LAMPIRAN

Lampiran 1.Surat Kesepakatan Bimbingan Skripsi

SURAT KESEPAKATAN BIMBINGAN TUGAS AKHIR

Kami yang bertanda tangan di bawah ini:

Pihak Pertama

Nama	: Indra Kusuma
NIM	: 21090042
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Ir. Ginanjar Wiro Sasmito, M.Kom., IPM., ASEAN Eng.
Status	: Dosen
NIDN	: 0613028601
Jabatan Fungsional	: Lektor Kepala
Pangkat/Golongan	: Penata Tk. I/ III D

Pada hari ini Selasa tanggal 15 April 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing I/II Tugas Akhir Pihak Pertama dengan syarat Pihak Pertama wajib melakukan bimbingan Tugas Akhir minimal 8 kali kepada Pihak Kedua. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Tugas Akhir

Tegal, 15 April 2025

Pihak Pertama,	Pihak Kedua,
Indra Kusuma	Ir. Ginanjar Wiro Sasmito, M.Kom., IPM., ASEAN Eng.

Mengetahui,
Ka. Prodi Sarjana Terapan Teknik Informatika

Dyah Ari Nugrahi, S.T., M.Kom
NIP. 09.015.225

1

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan dibawah ini:

Pihak Pertama

Nama	: Indra Kusuma
NIM	: 21090042
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Sharfina Febbi Handayani, S.kom., M.kom.
Status	: Dosen
NIDN	: 0617029201
Jabatan Fungsional	: Asisten Ahli
Pangkat/Golongan	: Penata muda tk. I/ III-b

Pada hari ini Rabu tanggal 13 Maret 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing II Skripsi Pihak Pertama dengan syarat Pihak Pertama wajib melakukan bimbingan Skripsi minimal 8 kali kepada Pihak Kedua. Adapun waktu dan tempat pelaksanaan disepakati antar pihak dilaksanakan hari rabu atau kamis pukul 08.00-12.00 wib dengan menunjukan progres.Pihak pertama menyatakan bersedia untuk menyelesaikan skripsi tepat waktu dan apabila pihak pertama tidak menyelesaikan skripsi tepat waktu sesuai dengan kespakatan maka pihak ke dua tidak akan memberikan rekomendasi ujian skripsi.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi.

Tegal, 13 Maret 2025

Pihak Pertama	Pihak Kedua
Indra Kusuma	Sharfina Febbi Handayani, S.Kom., M.Kom.

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

Dyah Apriliani, S.T., M.Kom.
NIPY. 09.015.225

Lampiran 2. Surat Pernyataan Pengajuan HKI

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini :

1. Nama : Indra Kusuma
   Kewarganegaraan : Indonesia
   Alamat : Desa Sigentong RT.01 RW.04, Kecamatan Wanasari, Kabupaten Brebes, Jawa Tengah, 52252

2. Nama : Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jln. Raya Kluwut Timur No. 24, Kecamatan Bulakamba, Kabupaten Brebes.

3. Nama : Sharfina Febbi Handayani, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Desa Grobog Kulon RT.02 RW.06, Kecamatan Pangkah Kabupaten Tegal. Provinsi Jawa Tengal. 52471, Pangkah Tegal

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:
   Berupa : Program Komputer
   Berjudul : Aplikasi Penjualan Dengan Optimalisasi Stok Barang Menggunakan K-Means Clustering Dan Integrasi Whatsapp Bot Pada Raka Store

   - Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);
   - Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
   - Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;
   - Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
   - Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
   - Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.

3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketa pidana dan/atau perdata di Pengadilan.

4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 3 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa:
   a. permohonan karya cipta yang saya ajukan dianggap ditarik kembali; atau
   b. Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.

c. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam berperkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap.

Demikian Surat pernyataan ini saya/kami buat dengan sebenarnya dan untuk dipergunakan sebagimana mestinya.

Tegal, 25 Juni 2025

(Indra Kusuma)

(Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.)

(Sharfina Febbi Handayani, S.Kom., M.kom.)

Lampiran 3. Surat pengalihan HKI

SURAT PENGALIHAN HAK CIPTA

Yang bertanda tangan di bawah ini :

1. Nama : Indra Kusuma
   Kewarganegaraan : Indonesia
   Alamat : Desa Sigentong RT.01 RW.04, Kecamatan Wanasari, Kabupaten Brebes, Jawa Tengah, 52252

2. Nama : Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jln. Raya Kluwut Timur No. 24, Kecamatan Bulakamba, Kabupaten Brebes.

3. Nama : Sharfina Febbi Handayani, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Desa Grobog Kulon RT.02 RW.06, Kecamatan Pangkah Kabupaten Tegal. Provinsi Jawa Tengal. 52471, Pangkah Tegal

Adalah **Pihak I** selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada :

Nama : Pusat Penelitian dan Pengabdian Masyarakat (P3M)
Alamat : Jl. Mataram No. 9 Pesurungan Lor Tegal

Adalah **Pihak II** selaku Pemegang Hak Cipta berupa Program Komputer dengan judul "Aplikasi Penjualan Dengan Optimalisasi Stok Barang Menggunakan K-Means Clustering Dan Integrasi Whatsapp Bot Pada Raka Store". untuk didaftarkan di Direktorat Hak Cipta dan Desain Industri, Direktorat Jenderal Kekayaan Intelektual, Kementerian Hukum dan Hak Asasi Manusia Republik Indonesia.

Demikianlah surat pengalihan hak ini kami buat, agar dapat dipergunakan sebagaimana mestinya.

Tegal, 25 Juni 2025

Pemegang Hak Cipta
Kepala P3M

(Muhammad Fikri Hidayatullah, S.T., M.Kom.)

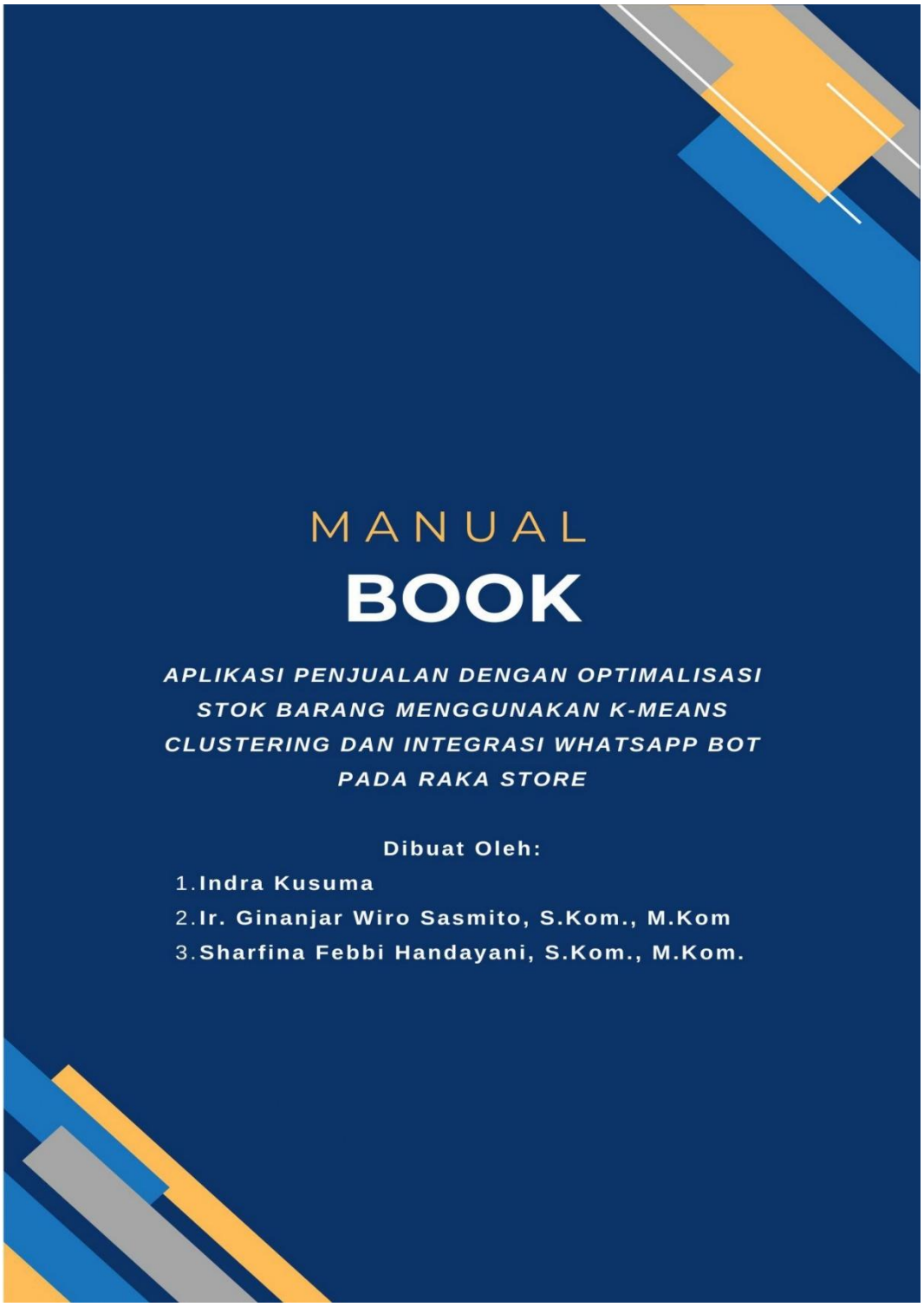
Pencipta

(Indra Kusuma)

(Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom)

((Sharfina Febbi Handayani, S.Kom., M.kom.)

Lampiran 4. *Manual Book dan Dokumen Teknikal*



1. PENDAHULUAN

1.1 Tujuan Pembuatan Dokumen

Dokumen ini dibuat untuk memberikan panduan penggunaan aplikasi Raka Store yang membantu pengguna dalam mengelola penjualan, stok barang, dan melakukan analisis optimalisasi stok berbasis metode K-Means Clustering serta integrasi pengiriman notifikasi melalui WhatsApp Bot.

1.2 Deskripsi Umum Sistem

Raka Store merupakan aplikasi berbasis web yang digunakan untuk mencatat transaksi penjualan, mengelola persediaan barang, melakukan analisis data penjualan, dan mempermudah komunikasi dengan pelanggan melalui WhatsApp.

1.3 Deskripsi Dokumen

Dokumen ini dibuat untuk memberikan panduan penggunaan "aplikasi raka store informasi sebagai berikut

1. BAB I.

Berisi informasi umum yang merupakan bagian pendahuluan, yang meliputi tujuan pembuatan dokumen, deskripsi umum sistem serta deskripsi dokumen.

2. BAB II.

Berisi perangkat yang dibutuhkan untuk penggunaan aplikasi meliputi perangkat lunak dan perangkat keras.

3. BAB III.

Berisi panduan penggunaan Aplikasi Raka Store

2. KEBUTUHAN SISTEM

2.1 Perangkat Lunak

- Sistem Operasi: Windows, Linux, atau macOS
- Browser: Google Chrome, Mozilla Firefox

2.2 Perangkat Keras

- Laptop/handpone
- Akses Internet

3. PANDUAN PENGGUNAAN

3.1 struktur menu

1. Menu Dashboard
2. Menu kategori
3. Menu produk
4. Menu stok

5. Menu pesanan
6. Menu penjualan
7. Menu Analisa produk
8. Menu laporam keuangan
9. Menu prediksi keuangan
10. Menu Kelola akun

3.2 Penggunaan

Pada bagian ini akan dijelaskan mengenai tata cara menggunakan Aplikasi Cutbox.

Aplikasi ini menawarkan berbagai fitur yang menarik dan sangat bermanfaat untuk membantu pengguna dalam mengelola produk, keuangan, dan analisis bisnis secara menyeluruh. Pada sistem ini, pengguna dapat mengakses berbagai menu dan modul yang dirancang untuk memberikan kemudahan dalam pengelolaan stok barang, pencatatan transaksi, hingga pembuatan laporan dan prediksi berbasis data. Fitur-fitur yang disediakan saling terintegrasi, sehingga memudahkan pengguna dalam mengawasi performa bisnis antara lain fitur – fiturnya sebagai berikut:

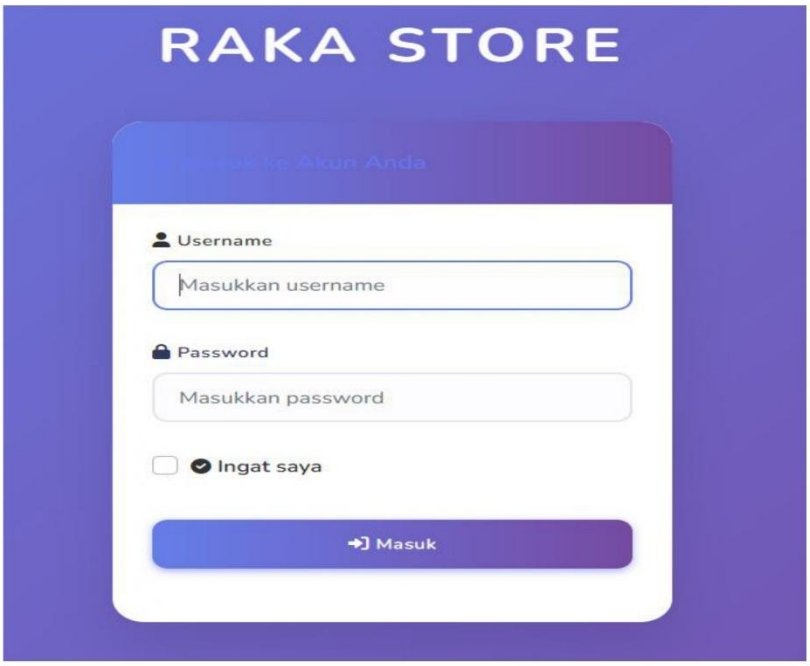
- Fitur Manajemen Kategori Produk
  Fitur ini memungkinkan pengguna untuk mengelola kategori produk yang tersedia dalam sistem. Pengguna dapat menambahkan kategori baru, mengedit kategori yang sudah ada, serta menghapus kategori yang tidak lagi digunakan. Setiap kategori dapat dilengkapi dengan nama dan deskripsi agar lebih informatif. Fitur ini sangat penting karena setiap produk harus dikaitkan dengan sebuah kategori, yang nantinya akan digunakan juga dalam pelaporan dan analisis performa produk. Sebelum penghapusan kategori, sistem juga akan mengecek apakah kategori tersebut masih digunakan oleh produk untuk mencegah kesalahan data.
- Fitur Manajemen Produk
  Fitur ini dirancang untuk membantu pengguna dalam mengelola data produk secara menyeluruh. Pengguna dapat menambahkan produk baru dengan detail lengkap seperti nama, kategori, harga beli, harga jual, satuan, stok minimal, dan gambar produk. Sistem secara otomatis akan menghasilkan kode produk berdasarkan kategori yang dipilih, sehingga memudahkan dalam pengelompokan dan pencatatan. Selain itu, pengguna juga dapat mengedit informasi produk atau menghapus produk yang tidak lagi digunakan, selama produk tersebut belum terkait dengan transaksi lain. Fitur ini menjadi dasar penting dalam mengelola inventaris toko secara efisien dan rapi.
- Fitur Penambahan Stok (Stock In) Fitur Penambahan Stok (Stock In)
  Fitur ini memungkinkan pengguna mencatat proses pemasukan stok baru dari supplier atau gudang secara rinci. Pengguna dapat memilih beberapa produk sekaligus, menentukan jumlah dan harga beli masing-masing, serta menambahkan keterangan dan sumber stok. Setiap transaksi penambahan stok akan secara otomatis memperbarui jumlah stok di sistem serta mencatat nilai total pembelian. Uniknya, setelah transaksi disimpan, sistem akan mengirimkan notifikasi otomatis melalui WhatsApp kepada pemilik toko menggunakan integrasi API Fonnte, sehingga informasi stok dapat tersampaikan secara cepat dan real-time.
- Fitur Manajemen Pesanan (Order)

Fitur pesanan atau transaksi penjualan merupakan inti dari proses bisnis. Di dalam sistem ini, pesanan dicatat melalui entri penjualan yang mencakup data pelanggan, daftar produk yang dibeli, jumlah, harga jual, dan total pembayaran. Semua transaksi penjualan akan langsung tercatat dalam laporan keuangan dan digunakan dalam perhitungan laba rugi, HPP, dan gross profit. Data ini juga menjadi sumber utama untuk fitur analisis dan prediksi. Selain itu, pengguna juga dapat melihat daftar transaksi sebelumnya, yang bisa difilter berdasarkan tanggal atau pelanggan.

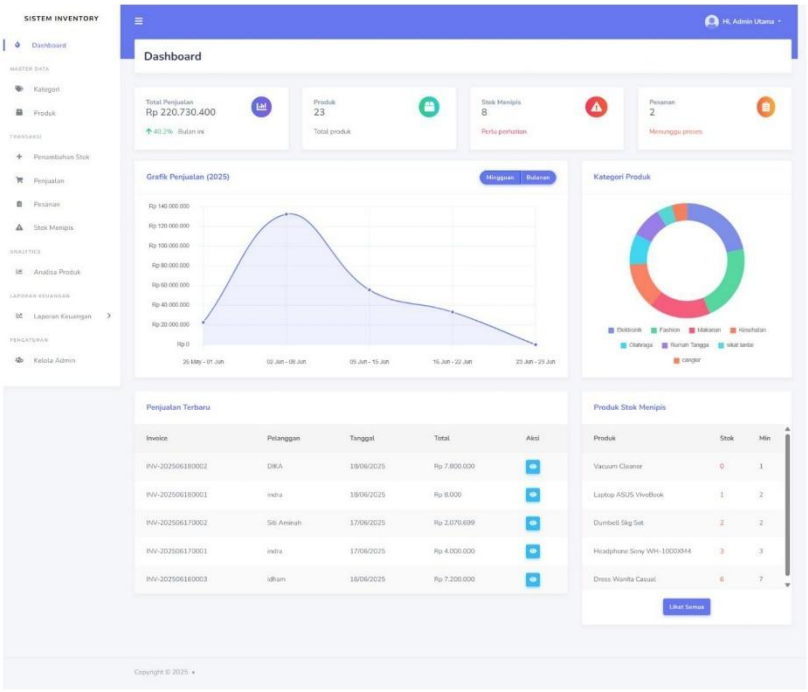
- Fitur Analisis Produk (Kinerja Penjualan)
  Fitur ini memberikan informasi mendalam mengenai performa masing-masing produk berdasarkan data penjualan aktual. Pengguna dapat mengetahui produk mana yang paling sering terjual, total pendapatan yang dihasilkan, serta margin keuntungan yang diperoleh. Informasi ini ditampilkan dalam bentuk laporan per produk dan kategori. Produk yang memiliki performa tinggi akan ditandai sebagai best performer, sedangkan produk yang memiliki stok tetapi tidak pernah terjual akan masuk dalam daftar produk kurang laku. Fitur ini membantu pengguna dalam mengevaluasi produk secara berkala dan menetapkan strategi promosi atau penghentian penjualan.
- Fitur Prediksi Keuangan
  Selain hanya menampilkan estimasi total pendapatan dan pengeluaran, fitur prediksi keuangan ini juga menyediakan detail prediksi per bulan dalam bentuk visualisasi grafik tren. Prediksi mencakup proyeksi laba, estimasi pengeluaran tetap dan variabel, serta estimasi akurasi model yang digunakan. Sistem juga menyajikan ringkasan insight seperti tren peningkatan penjualan atau potensi penurunan margin di bulan-bulan tertentu. Bahkan tersedia fitur debug untuk membantu developer atau admin mengevaluasi ketepatan data prediksi. Fitur ini memberikan dukungan strategis berbasis data untuk mengelola risiko keuangan dan peluang pertumbuhan bisnis.
- Fitur Laporan keuangan
  Fitur ini menyediakan ringkasan keuangan dalam bentuk laporan laba rugi berdasarkan periode waktu tertentu. Laporan mencakup total pendapatan yang diperoleh dari penjualan serta total biaya pokok penjualan (HPP) yang dihitung dari jumlah barang terjual dikalikan harga beli masing-masing produk. Selisih dari keduanya menghasilkan laba kotor atau gross profit. Selain itu, tersedia juga data bulanan untuk menunjukkan tren pendapatan, HPP, dan laba kotor dari bulan ke bulan selama periode yang dipilih. Fitur ini sangat berguna bagi pemilik usaha untuk memahami kondisi keuangan dasar perusahaan secara menyeluruh.

3.2.1 akses web

a. Ketikkan alamat https://rakastore.com
Kemudian masukan username dan password kemudian tekan masuk (pastikan tersambung ke internet) seperti yang terlihat pada gambar 3.1



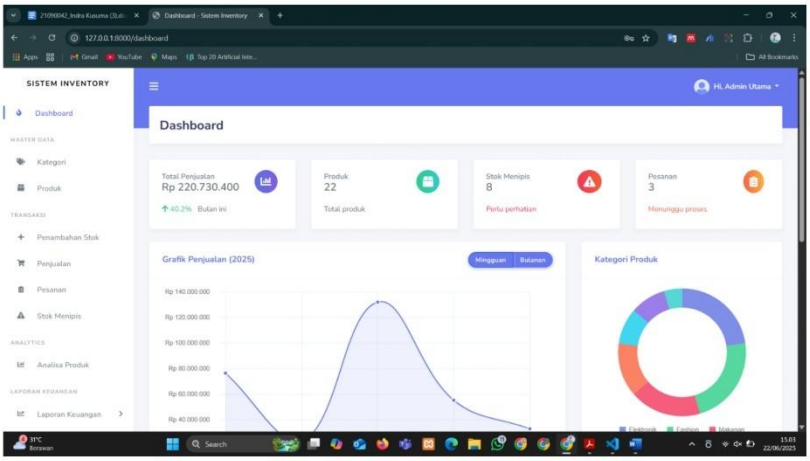
Gambar 3. 1 halaman login



Gambar 3. 2 *halaman dashboard*

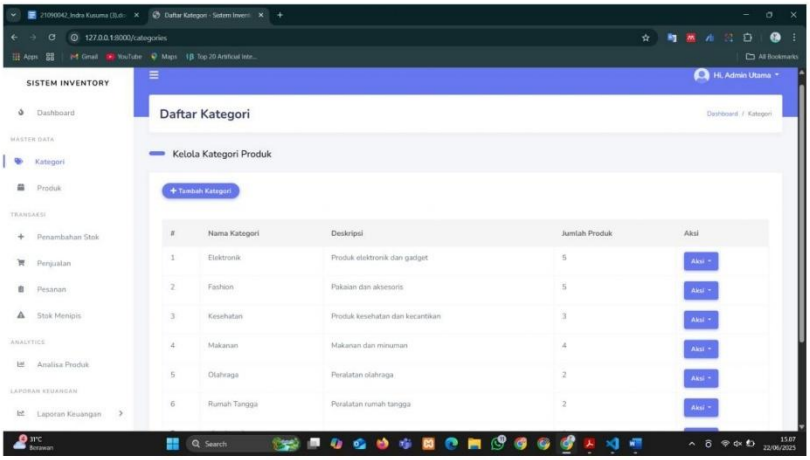
b. Jika berhasil, makan akan diarahkan ke halaman dashboard , dan akan ditampilkan total penjualan, grafik penjualan mingguan dan bulanan, jumlah total produk, daftar produk dengan stok menipis, data pesanan tertunda, grafik distribusi kategori produk, serta tabel data penjualan terbaru seperti yang telihat pada gambar 3.2

3.2.2 Menu kategori

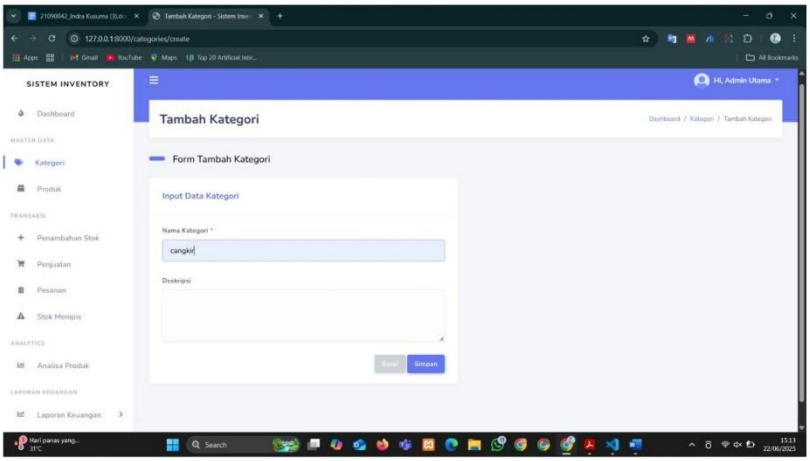


Gambar 3. 3 *side bar*

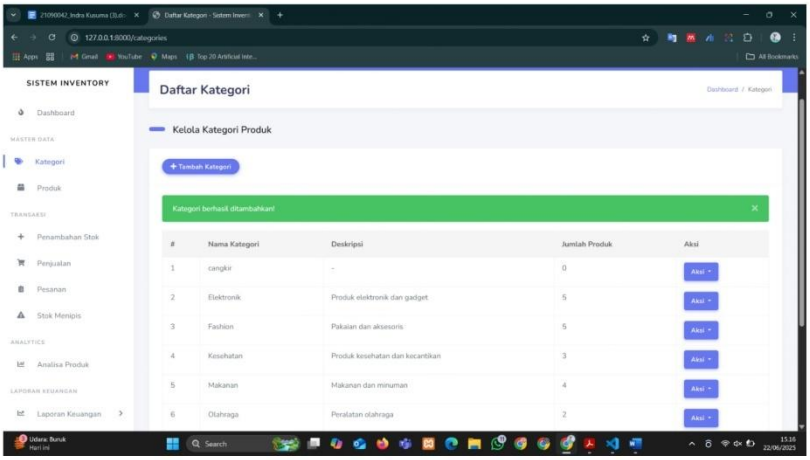
a. Untuk masuk ke Data Kategori, silakan klik menu "Kategori" seperti terlihat pada gambar berikut ini. Setelah masuk ke halaman daftar kategori, klik tombol "Tambah Kategori Baru", kemudian isi nama kategori (wajib), tambahkan deskripsi jika diperlukan, lalu klik "Simpan" untuk menyimpan data kategori yang baru dibuat. seperti yang telihat pada gambar 3.3 sampai 3.5



Gambar 3. 4 menu kategori

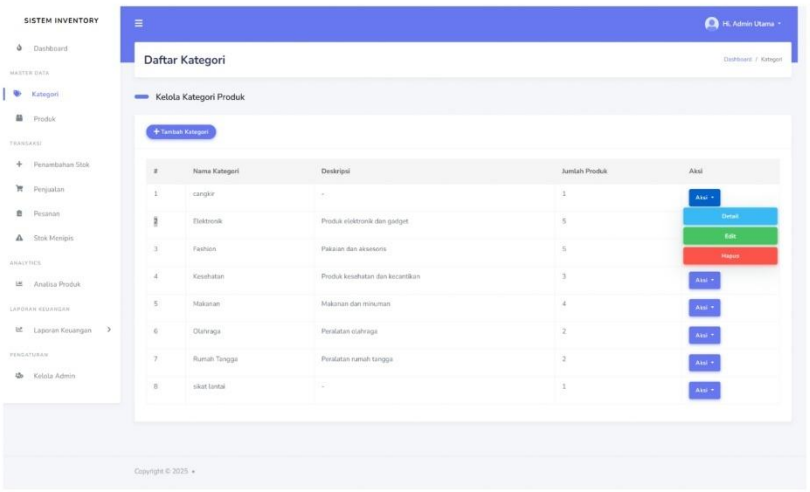


Gambar 3. 5 menu tambah kategori

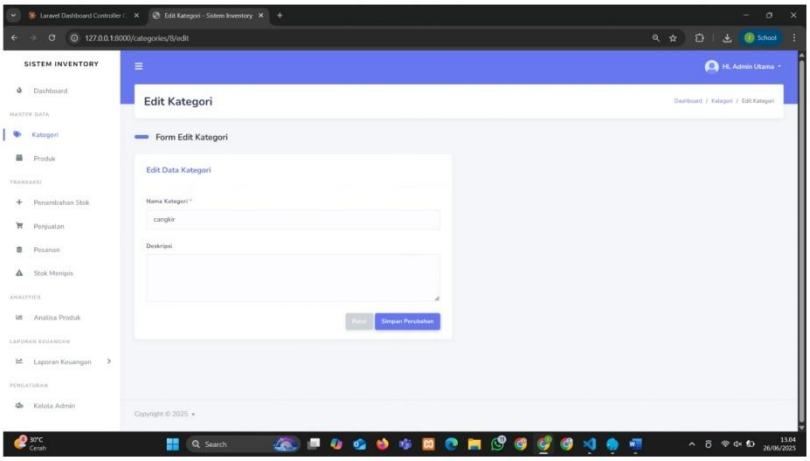


Gambar 3. 6 notifikasi penambahan produk

b. Untuk mengakses halaman edit kategori, silakan tekan menu "Edit" pada data yang diinginkan, maka akan muncul form dengan data yang sudah terisi, yaitu kolom Nama Kategori yang berisi nama saat ini dan kolom Deskripsi yang berisi deskripsi yang ada (jika tersedia). Selanjutnya, lakukan perubahan pada kolom yang diinginkan, pastikan Nama Kategori tetap diisi karena bersifat wajib, isi Deskripsi jika diperlukan, lalu tekan tombol "Simpan Perubahan" untuk menyimpan hasil edit seperti yang telihat pada gambar 3.7 sampai 3.8



Gambar 3. 7 pilihan tombol aksi

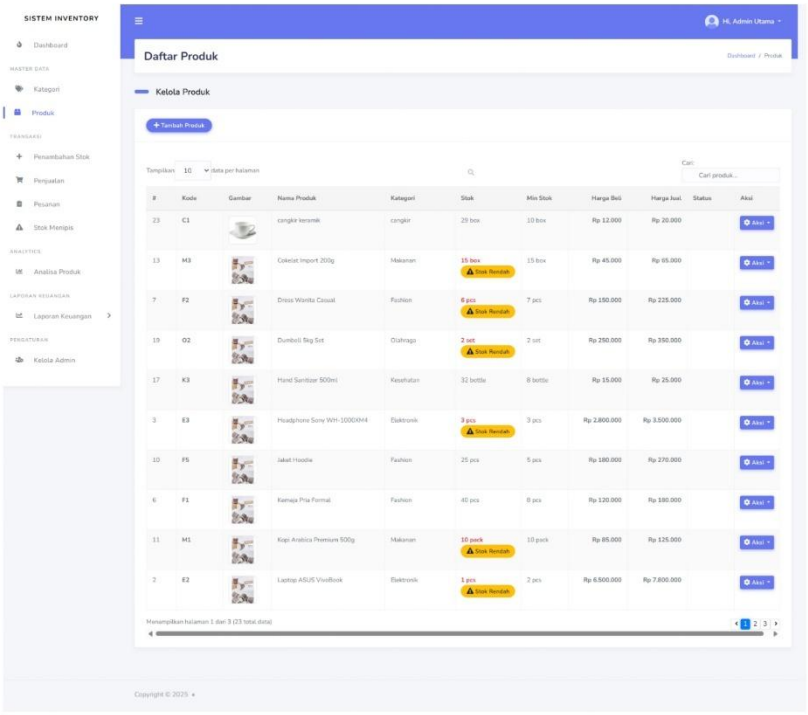


Gambar 3. 8 menu edit kategori

c. Untuk menghapus kategori, silakan tekan tombol "hapus" pastikan data kategori sedang tidak digunakan di data detail produk seperti yang terlihat pada gambar 3.7

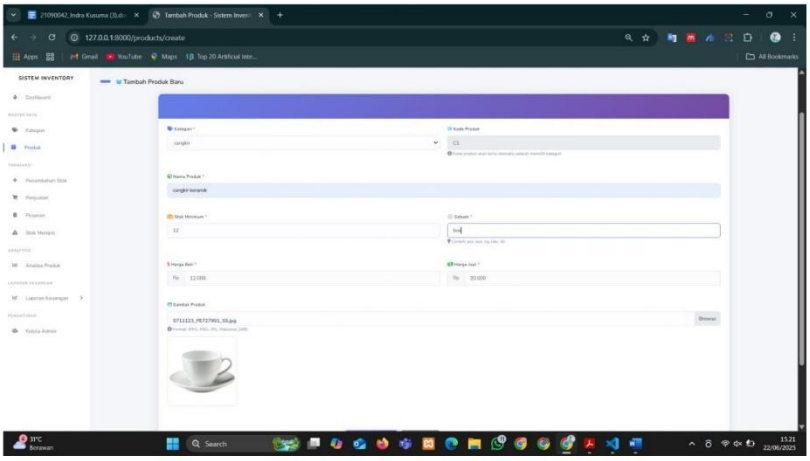
3.2.3 menu produk

a. Untuk membuka halaman produk, silakan tekan menu "Produk", lalu akan ditampilkan data produk yang mencakup kode produk, gambar produk, nama produk, nama kategori, jumlah stok produk, batas minimal stok, harga beli dan harga jual, serta tombol aksi yang dapat digunakan untuk mengelola data produk tersebut seperti yang telihat pada gambar 3.9

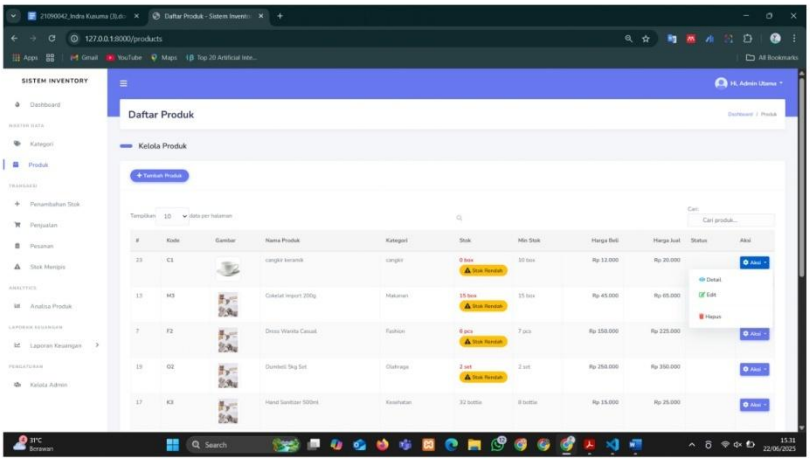


Gambar 3. 9 menu produk

b. Untuk menambahkan data barang, klik tombol "Tambah Produk Baru" dari halaman daftar produk, lalu isi nama produk, pilih kategori (kode produk akan terisi otomatis), lengkapi semua kolom yang tersedia, unggah gambar produk, dan klik tombol "Simpan" untuk menyimpan data produk tersebut seperti yang telihat pada gambar 3.9 sampai 3.10

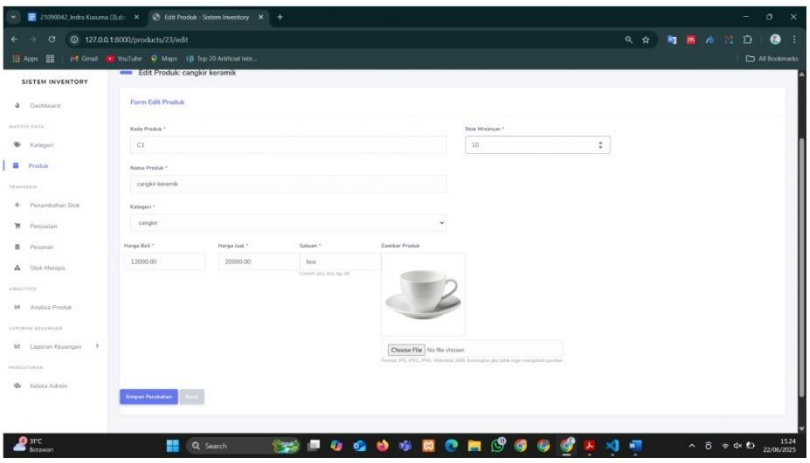


Gambar 3. 10 menu tambah produk

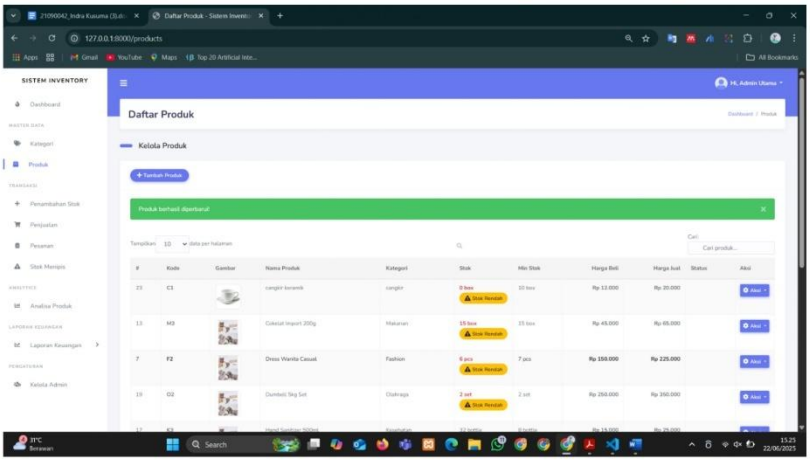


Gambar 3. 11 pilihan tombol aksi

c. Untuk mengedit data barang, klik tombol "Edit" pada data barang yang ingin diubah, lakukan perubahan pada data sesuai kebutuhan, lalu klik "Simpan Perubahan" untuk menyimpan hasilnya. Sementara itu, untuk melihat detail barang, silakan tekan tombol "Detail" pada menu dropdown di bagian aksi seperti yang telihat pada gambar 3.11 samapi 3.12

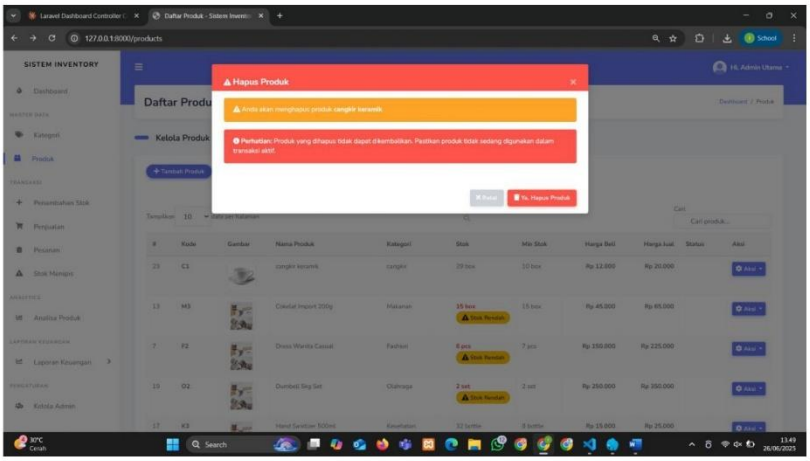


Gambar 3. 12 menu edit produk



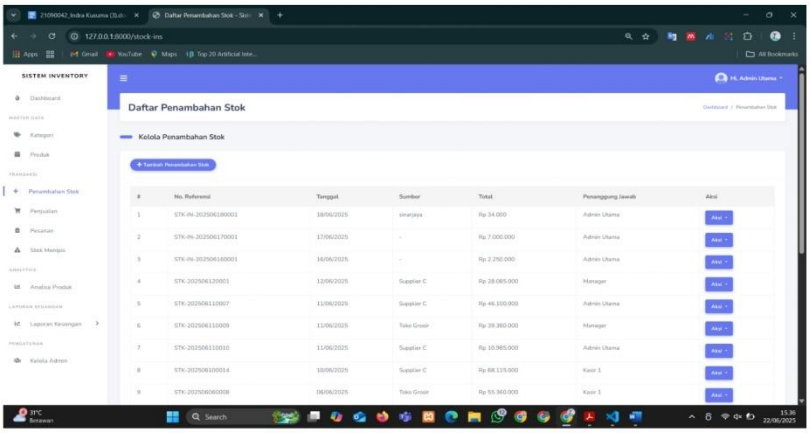
Gambar 3. 13 notifikasi perubahan data

d. Untuk menghapus data bisa menekan menu aksi dan pilih pilhan hapus dan akan mucul pop up button dan pilih hapus seperti yang telihat pada gambar 3.14



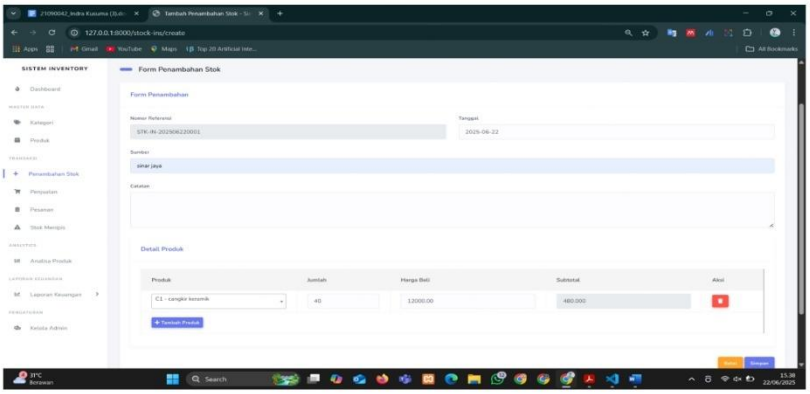
Gambar 3. 14 *pop up button hapus*

3.2.4 menu stok produk



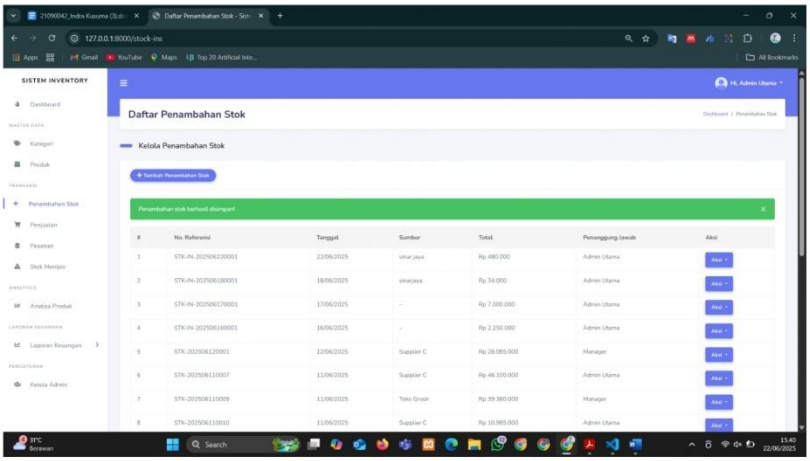
Gambar 3. 15 menu stok

a. Untuk membuka halaman penambahan stok, silakan klik menu "Penambahan Stok". Setelah masuk ke menu Stock In, Anda akan melihat data yang ditampilkan seperti nomor referensi sebagai kode unik setiap transaksi, tanggal penambahan stok, nama supplier atau sumber stok, total nilai penambahan stok, nama staff yang memproses, serta tombol aksi untuk melihat detail, mengedit, atau menghapus data penambahan stok tersebut. seperti yang telihat pada gambar 3.15.



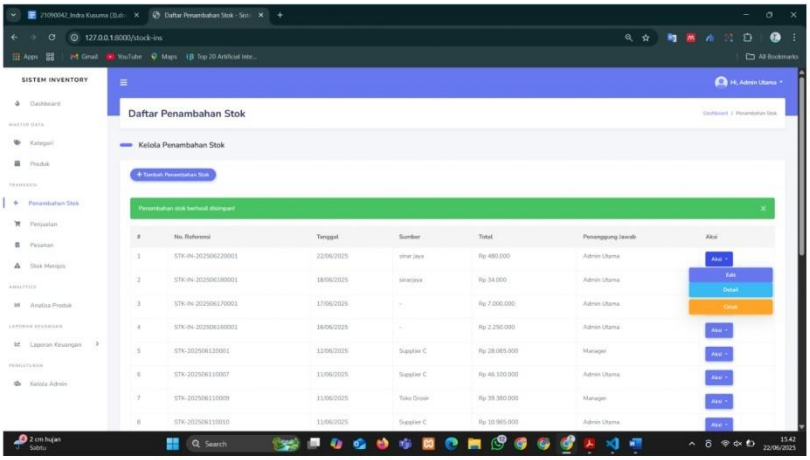
Gambar 3. 16 halaman penambahan stok

b. Untuk menambah stok baru, dari halaman daftar Stock In, klik tombol "Tambah Stok", maka Anda akan diarahkan ke halaman form penambahan stok. Pada halaman tersebut, akan ditampilkan beberapa form seperti nomor referensi yang terisi otomatis, tanggal penambahan, sumber supplier (opsional), dan catatan (opsional). Untuk menambahkan produk, tersedia form pemilihan produk yang sudah terhubung dengan data produk, sehingga Anda hanya perlu memasukkan jumlah produk yang ditambahkan, lalu tekan tombol "Simpan" untuk menyimpan data penambahan stok. gambar 3.15 sampai 3. 16

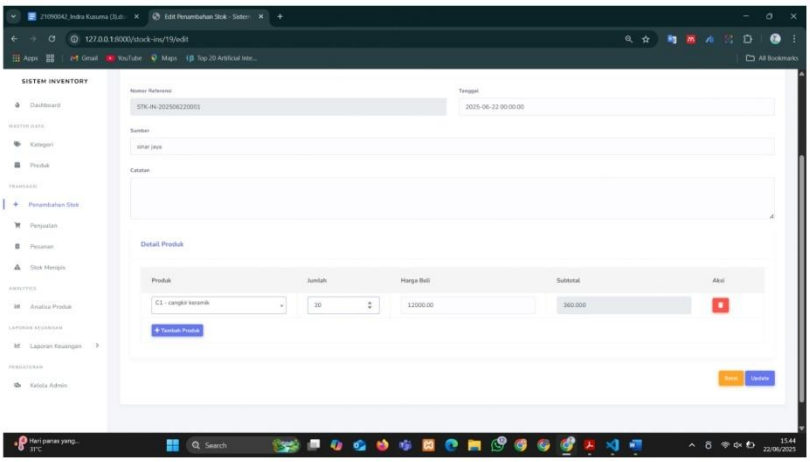


Gambar 3. 17 notifikasi penambahan stok

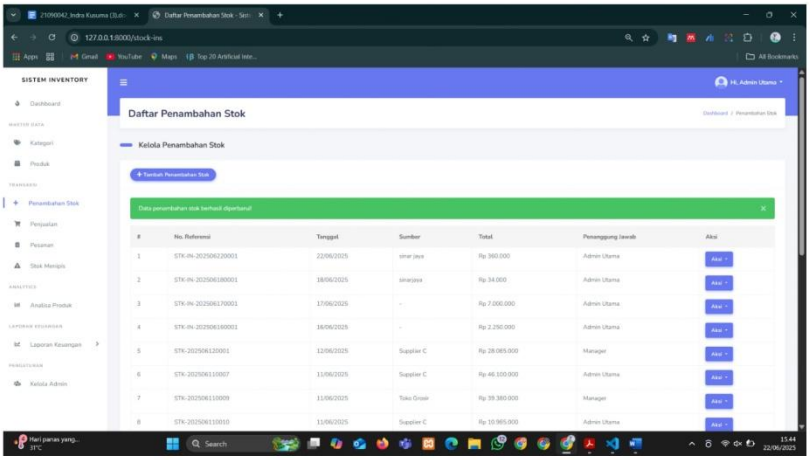
c. Untuk mengedit data stok barang, klik tombol "Edit" pada menu aksi, kemudian Anda akan diarahkan ke halaman form edit. Form edit ini memiliki tampilan yang sama dengan form penambahan stok, namun sudah terisi dengan data sebelumnya. Setelah melakukan perubahan yang diperlukan, klik tombol "Update" untuk menyimpan perubahan data stok. gambar 3.18 sampai 3. 19



Gambar 3. 18 pilihan menu aksi

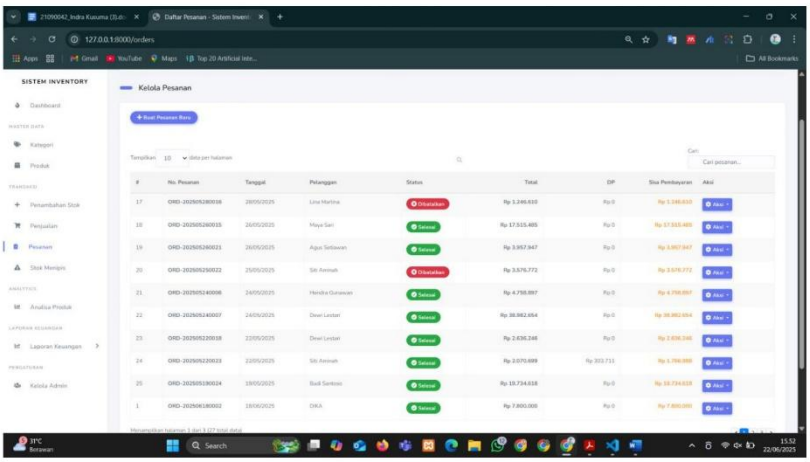


Gambar 3. 19 halaman menu edit stok



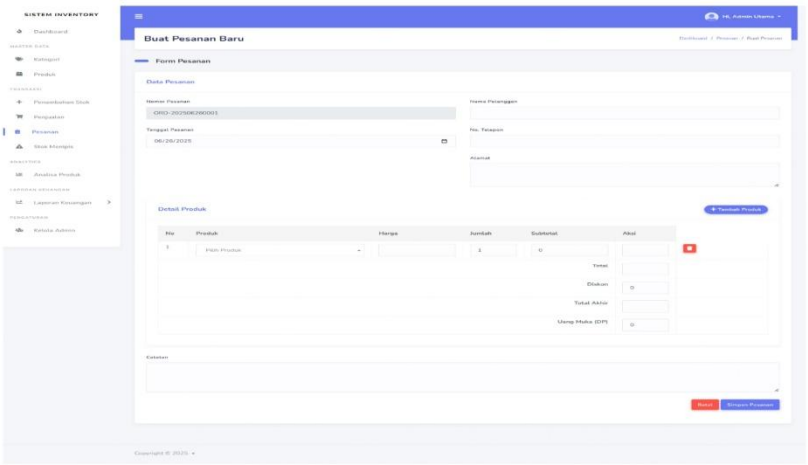
Gambar 3. 20 notifikasi perubahan data

3.2.5 menu pesanan



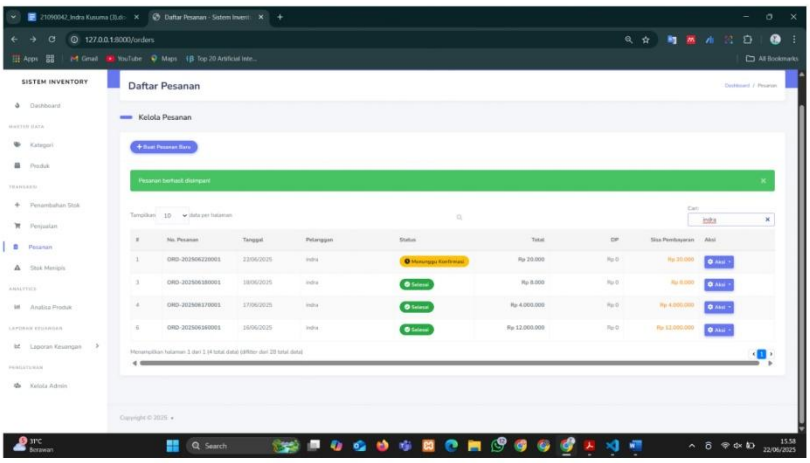
Gambar 3. 21 menu pesanan

a. Untuk membuka halaman pesanan, silakan klik menu "Pesanan". Setelah masuk ke halaman tersebut, Anda akan melihat informasi seperti nomor pesanan, nama pelanggan, tanggal pesanan, status pesanan (Pending, Completed, atau Cancelled), dan total nilai pesanan. Selain itu, tersedia tombol aksi dengan berbagai pilihan, yaitu melihat detail pesanan, mengedit pesanan, menghapus pesanan, mencetak pesanan, memproses pesanan menjadi penjualan, serta membatalkan pesanan. Tersedia juga kolom pencarian untuk memudahkan dalam menemukan pesanan tertentu gambar 3.21



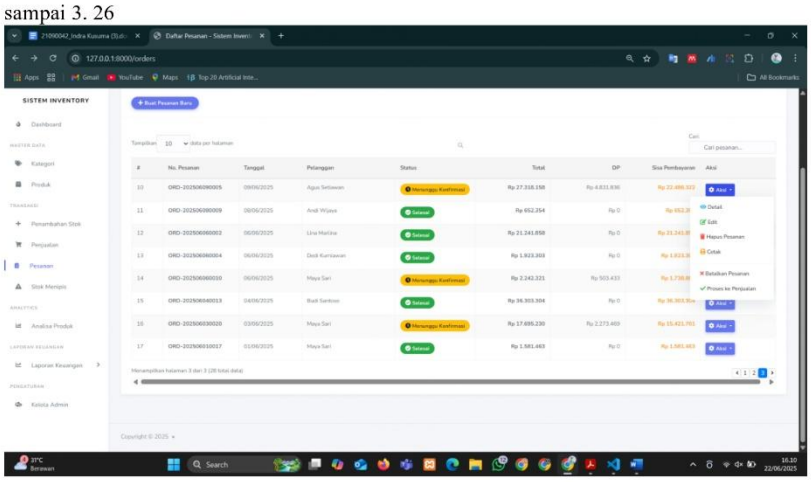
Gambar 3. 22 halaman tambah pesanan

b. Untuk menambah pesanan baru, klik tombol "Tambah Pesanan Baru", lalu Anda akan diarahkan ke halaman form tambah pesanan. Pada form ini, beberapa kolom yang harus diisi antara lain: Nomor Pesanan yang otomatis tergenerate, Nama Pelanggan, Tanggal Pesanan, serta kolom opsional seperti Nomor Telepon, Alamat Pelanggan, Uang Muka, dan Catatan. Setelah mengisi data pelanggan, langkah selanjutnya adalah menambahkan produk yang akan dipesan dengan memilih produk dari dropdown yang tersedia, kemudian klik tombol "Tambah Item" untuk menambahkan baris produk baru. Anda dapat menambahkan beberapa produk sekaligus dalam satu pesanan, lalu tentukan jumlah masing-masing produk yang dipesan. Total harga akan terakumulasi secara otomatis, dan setelah semua data lengkap, tekan tombol "Simpan Pesanan" untuk menyimpan transaksi tersebut. Seperti yang terlihat pada gambar 3.21 sampai 3. 22

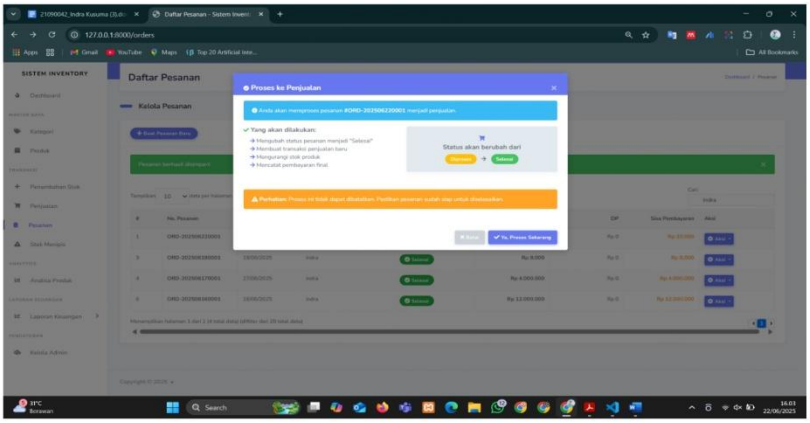


Gambar 3. 23 notifikasi penambahan pesanan

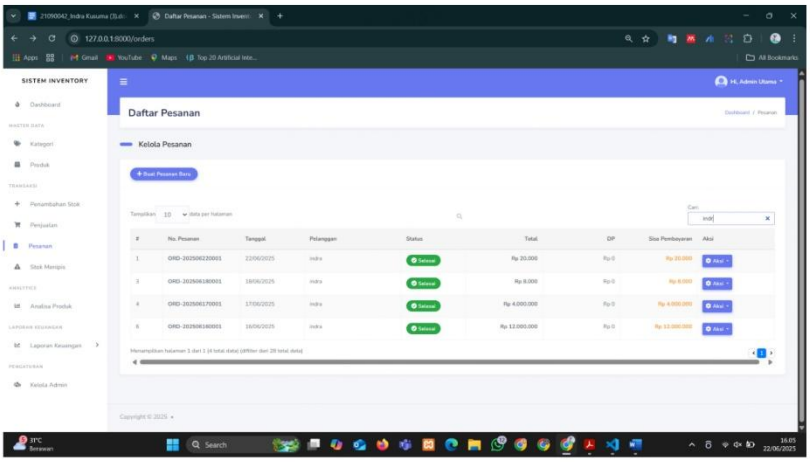
c. Untuk memproses pesanan, langkah pertama yang harus dilakukan adalah memastikan status pesanan berada dalam kondisi "Pending". Setelah itu, sistem akan secara otomatis memverifikasi ketersediaan stok produk. Jika stok tersedia, klik tombol "Proses ke Penjualan" pada pesanan yang dimaksud. Selanjutnya, sistem akan menampilkan kotak konfirmasi untuk memastikan tindakan tersebut. Setelah dikonfirmasi, pesanan akan secara otomatis dikonversi menjadi data penjualan dan proses pun selesai gambar 3.24 sampai 3. 26



Gambar 3. 24

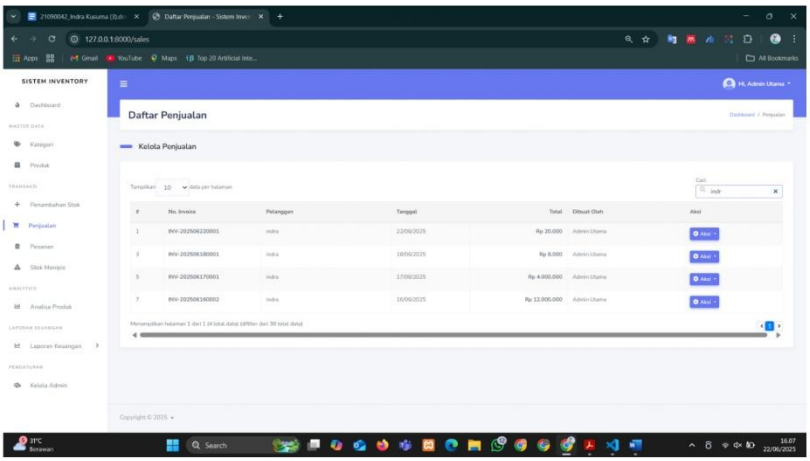


Gambar 3. 25 tombol pop up proses data ke penjualan



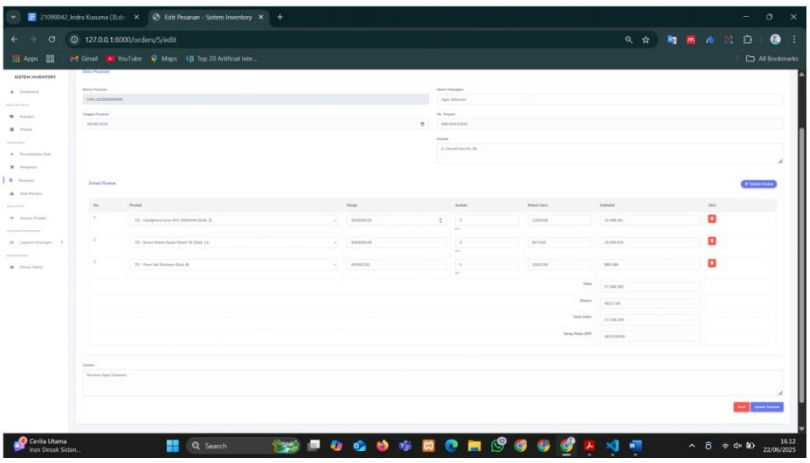
Gambar 3. 26 menu pesanan

d. untuk melihat data penjualan bisa klik menu penjualan

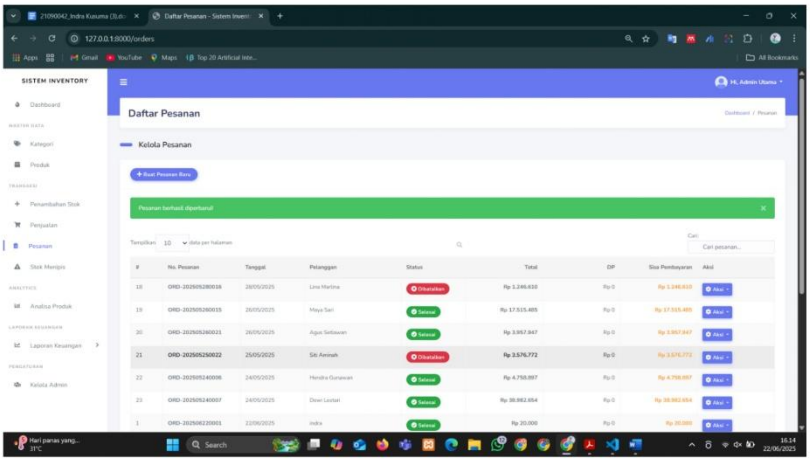


Gambar 3. 27 menu searching pesanan

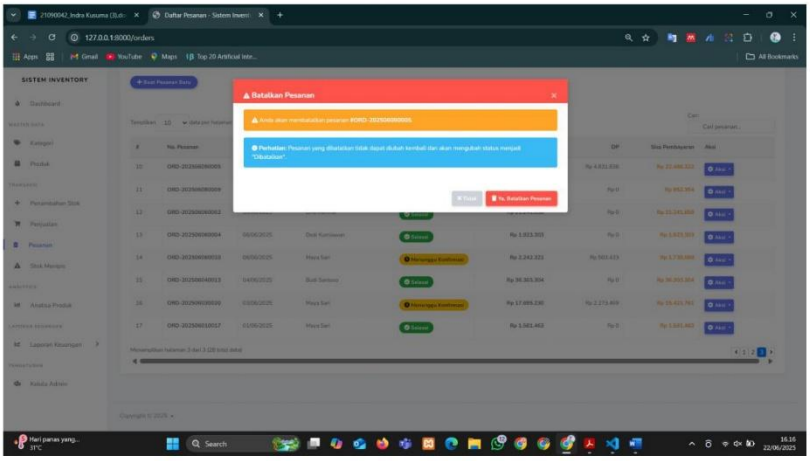
e. Untuk mengedit data pesanan, pastikan terlebih dahulu bahwa status pesanan masih dalam kondisi "Menunggu Konfirmasi", kemudian klik tombol "Edit". Setelah itu, isi form sesuai perubahan yang diperlukan, lalu klik tombol "Update Pesanan" untuk menyimpan pembaruan gambar 3.28 sampai 3. 29



Gambar 3. 28 halaman edit pesanan

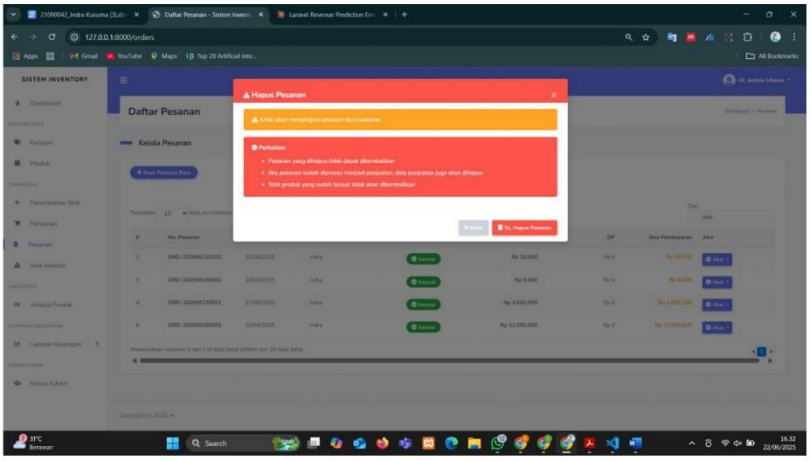


Gambar 3. 29 notifikasi perubahan data



Gambar 3. 30 tombol pop up pembatalan pesanan

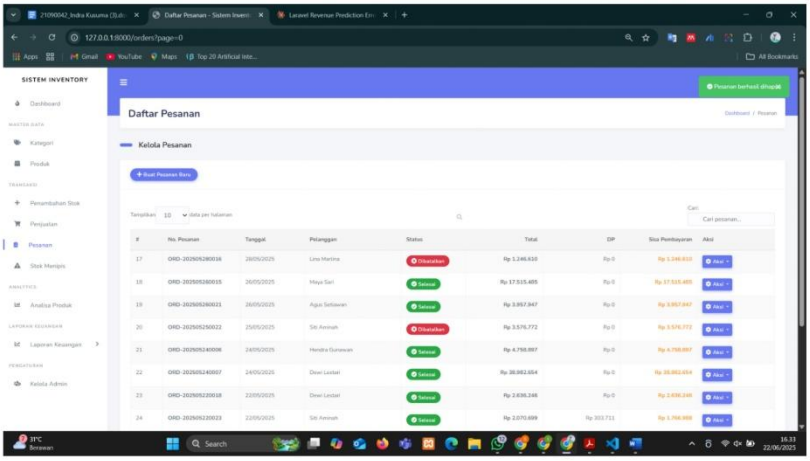
f. Untuk membatalkan pesanan, klik tombol "Batal" pada pesanan yang ingin dibatalkan, lalu sistem akan menampilkan kotak konfirmasi. Setelah konfirmasi dilakukan, status pesanan akan otomatis berubah menjadi "Cancelled" atau dibatalkan yang terlihat pada gambar 3.21 sampai 3. 30



Gambar 3. 31 tombol pop up menghapus pesanan

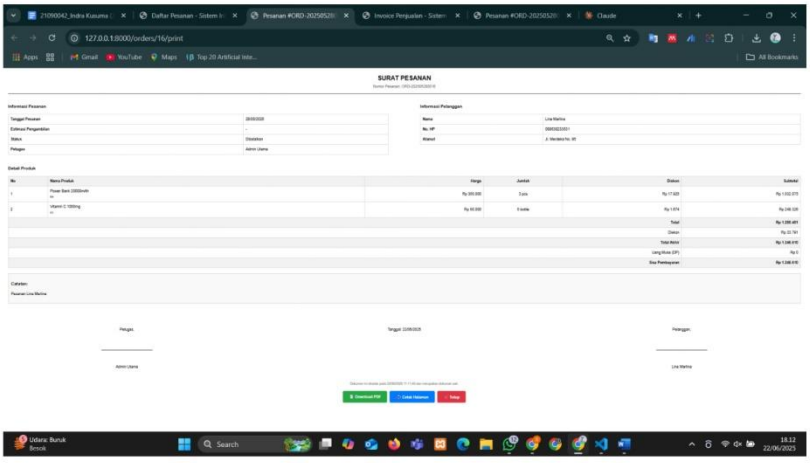
g. Untuk menghapus pesanan, klik tombol "Hapus" pada pesanan yang ingin dihapus, kemudian lakukan konfirmasi penghapusan saat sistem menampilkan peringatan. Harap diperhatikan bahwa menghapus pesanan akan menghapus seluruh data terkait

secara permanen, termasuk data penjualan jika pesanan tersebut sudah diproses yang terlihat pada gambar 3. 31



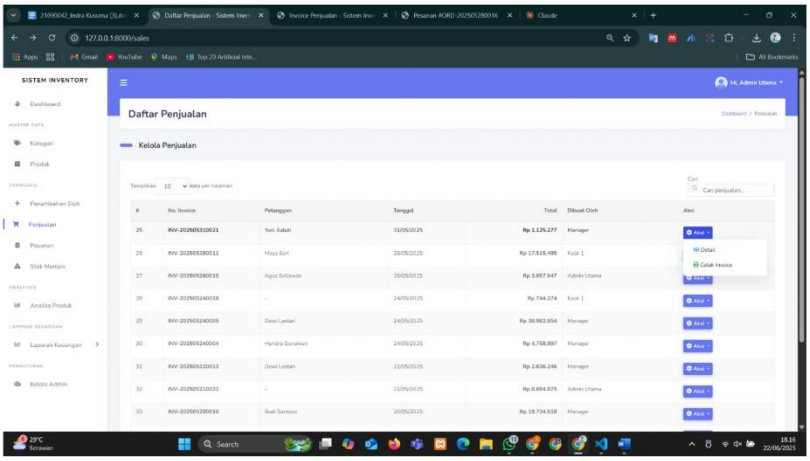
Gambar 3. 32 notifikasi

h. untuk cetak pesanan bisa menekan tombol cetak dan dapat menekan tombol download pdf untuk mendownload file struk pesanan yang terlihat pada gambar 3.33



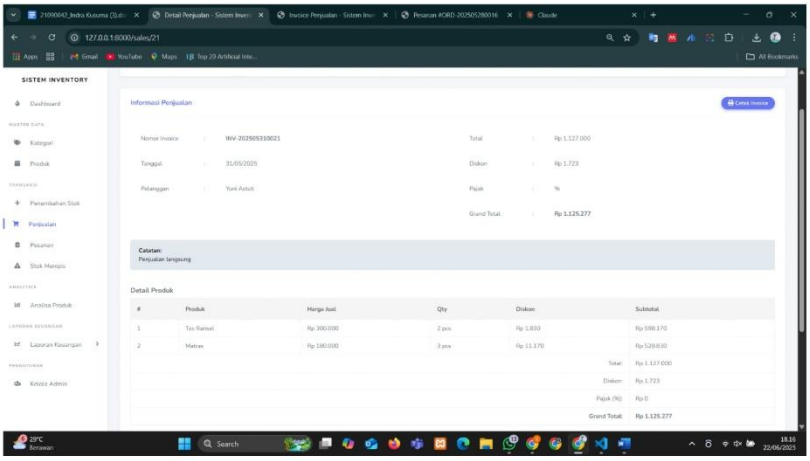
Gambar 3. 33 menu invoice pesanan

3.2.6 menu penjualan



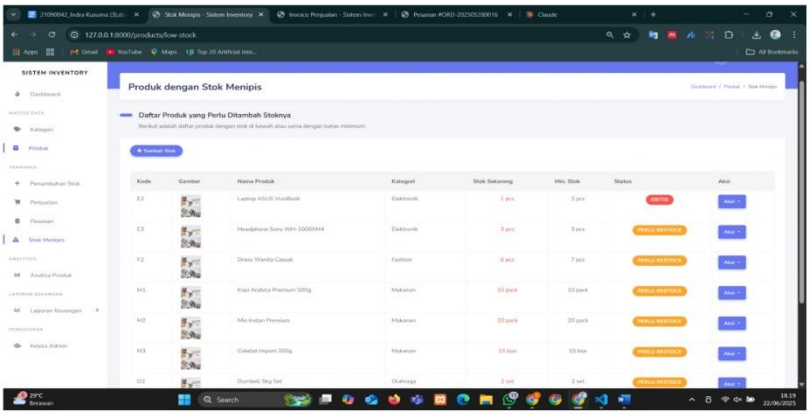
Gambar 3. 34 tombol aksi penjualan

a. Untuk menu penjualan ada 2 menu yaitu cetak struk penjualan dan melihat detail penjualan
b. Untuk melihat detail pesanan, klik tombol "Detail" pada pesanan yang ingin Anda tinjau. Setelah itu, sistem akan menampilkan informasi lengkap yang mencakup data pelanggan, rincian setiap produk yang dipesan, perhitungan total harga beserta diskon jika ada, serta riwayat status pesanan sejak dibuat hingga saat ini yang terlihat pada gambar 3.34 sampai 3. 35



Gambar 3. 35 menu invoice penjualan

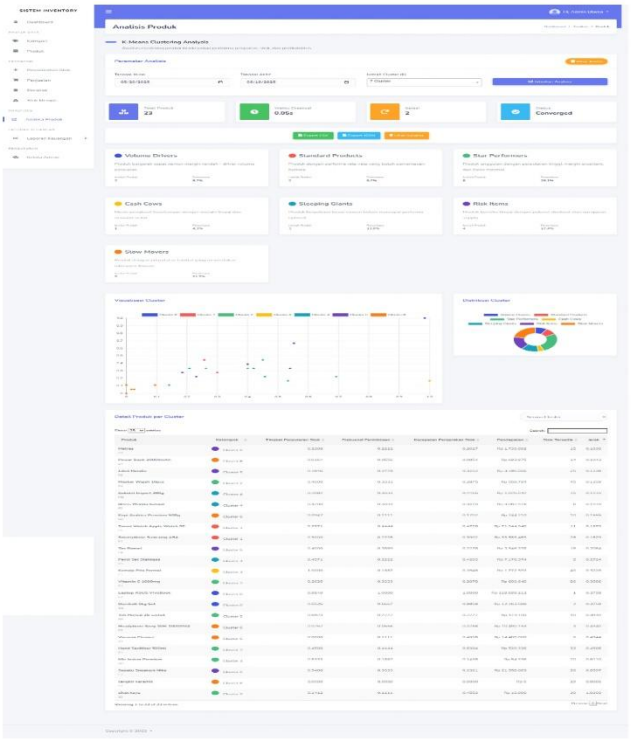
3.2.7 menu pemberitahuan stok menipis



Gambar 3. 36 menu stok menipis

a. Halaman ini dapat diakses dengan menekan menu "Stok Menipis". Pada halaman tersebut, sistem akan menampilkan laporan produk-produk yang stoknya menipis atau bahkan telah habis, sehingga memudahkan pengguna untuk segera melakukan tindakan restok yang terlihat pada gambar 3.36

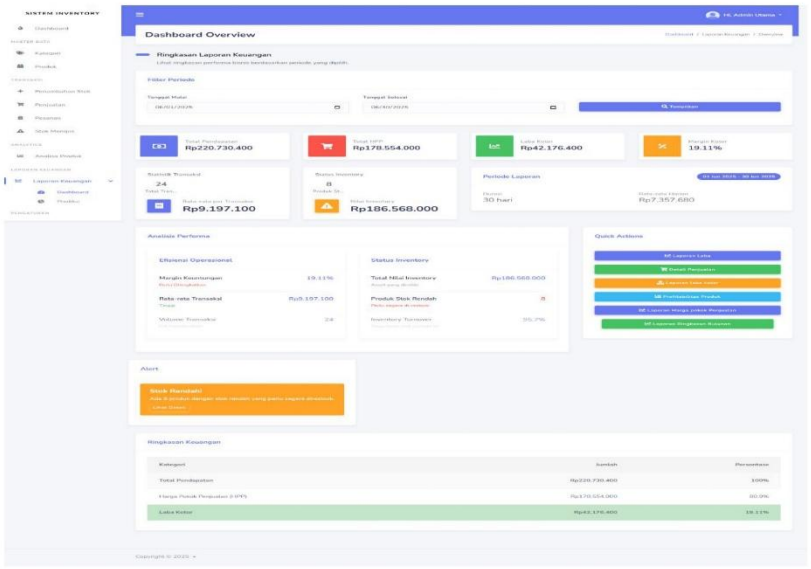
3.2.8 menu Analisis Produk



Gambar 3. 37 menu Analisa produk

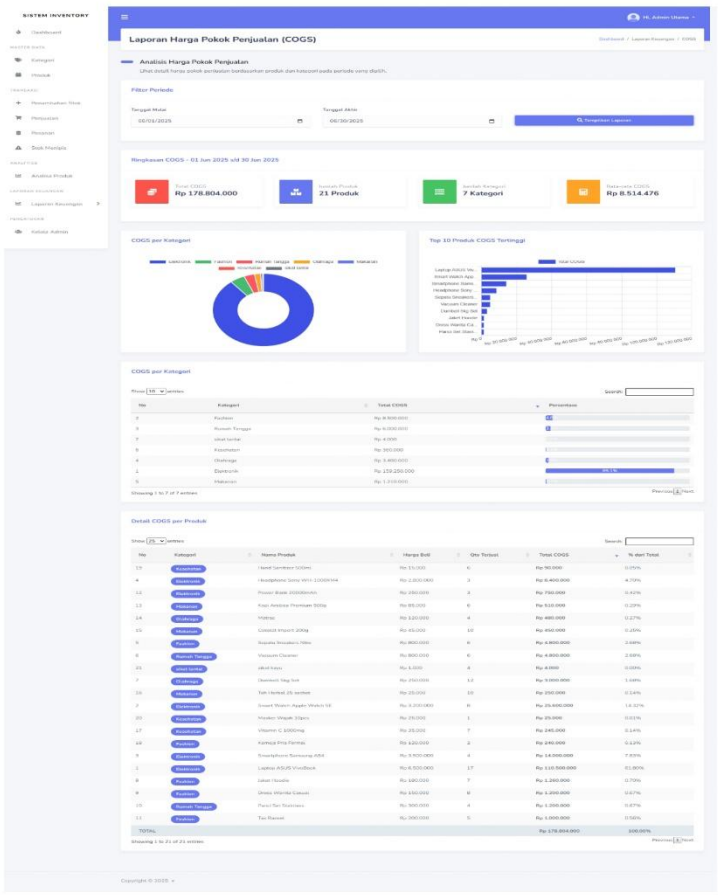
a. Halaman analisis produk dapat diakses dengan menekan menu "Analisa Produk". Untuk menggunakan fitur ini, pengguna perlu mengatur parameter analisis terlebih dahulu. Parameter utama yang harus ditentukan meliputi periode analisis, yaitu dengan memilih tanggal mulai (Start Date) dan tanggal akhir (End Date), dengan catatan bahwa rentang waktu maksimal adalah 1 tahun. Selain itu, pengguna juga perlu menentukan jumlah cluster yang diinginkan, dengan pilihan antara 3 hingga 7 cluster. Setelah semua parameter diisi, klik tombol "Jalankan Analisis" atau "Run Analysis". Sistem akan melakukan validasi terhadap input yang diberikan, dan jika valid, proses analisis akan segera dimulai. Selama proses berlangsung, progress akan ditampilkan secara real-time, dan estimasi waktu analisis berkisar antara 1 hingga 5 menit tergantung pada jumlah data yang dianalisis yang terlihat pada gambar 3.37

3.2.9 menu laporan keuangan



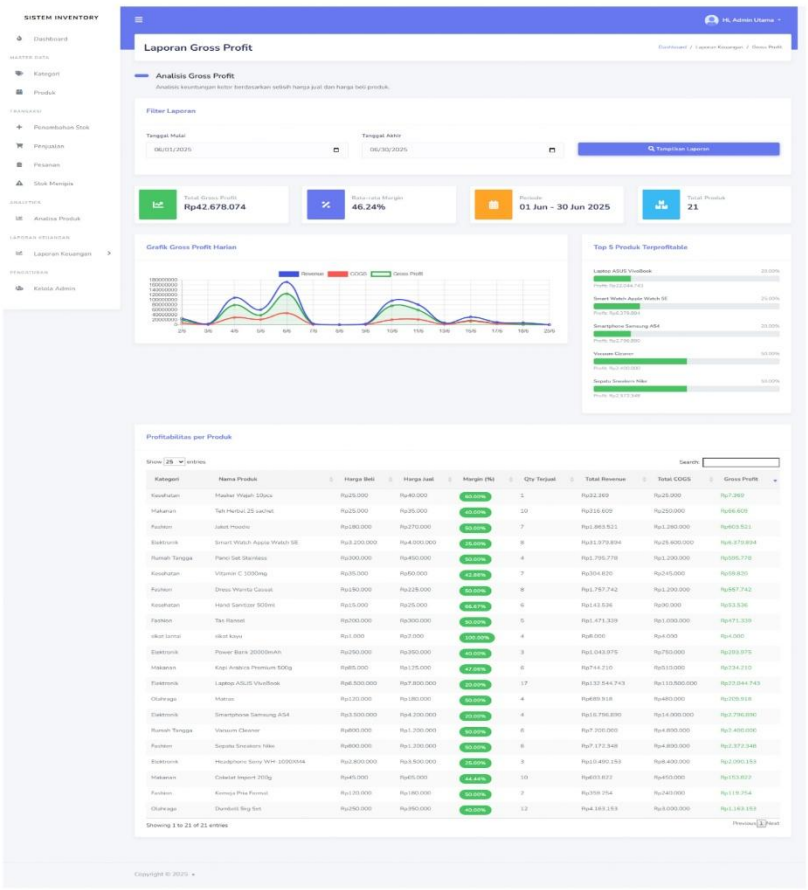
Gambar 3. 38 menu dashboard laporan keuangan

a. Menu-menu laporan keuangan dapat diakses melalui menu "Laporan Keuangan", lalu pilih submenu "Dashboard". Di dalamnya tersedia berbagai fitur utama yang dirancang untuk memberikan wawasan menyeluruh terhadap kondisi keuangan bisnis. Fitur-fitur tersebut meliputi: Laporan Laba Rugi yang menganalisis pendapatan dan beban usaha; Laporan COGS (Cost of Goods Sold) yang menampilkan rincian harga pokok penjualan; Analisis Gross Profit untuk melihat profitabilitas kotor; Detail Penjualan yang menyajikan analisis penjualan secara komprehensif; Laporan Inventory yang memperlihatkan status stok dan nilai persediaan; Profitabilitas Produk untuk menghitung return on investment (ROI) dari tiap produk; Dashboard Overview sebagai ringkasan menyeluruh dari semua metrik penting; serta Ringkasan Bulanan yang menampilkan analisis keuangan per bulan secara terstruktur.
b. Untuk menggunakan fitur laporan, langkah pertama adalah memilih jenis laporan yang ingin dilihat sesuai kebutuhan, seperti Laporan Laba Rugi, COGS, atau Detail Penjualan. Selanjutnya, pengguna dapat memanfaatkan fitur filter periode untuk menyesuaikan rentang waktu data yang ditampilkan. Semua jenis laporan mendukung filter periode dengan format tanggal sebagai berikut: Start Date (Tanggal Mulai) dan End Date (Tanggal Akhir), menggunakan format penulisan YYYY-MM-DD. Secara default, sistem akan menampilkan laporan untuk bulan berjalan, yaitu dari tanggal 1 hingga 31 pada bulan tersebut, jika pengguna tidak mengatur filter secara manual yang terlihat pada gambar 3.38



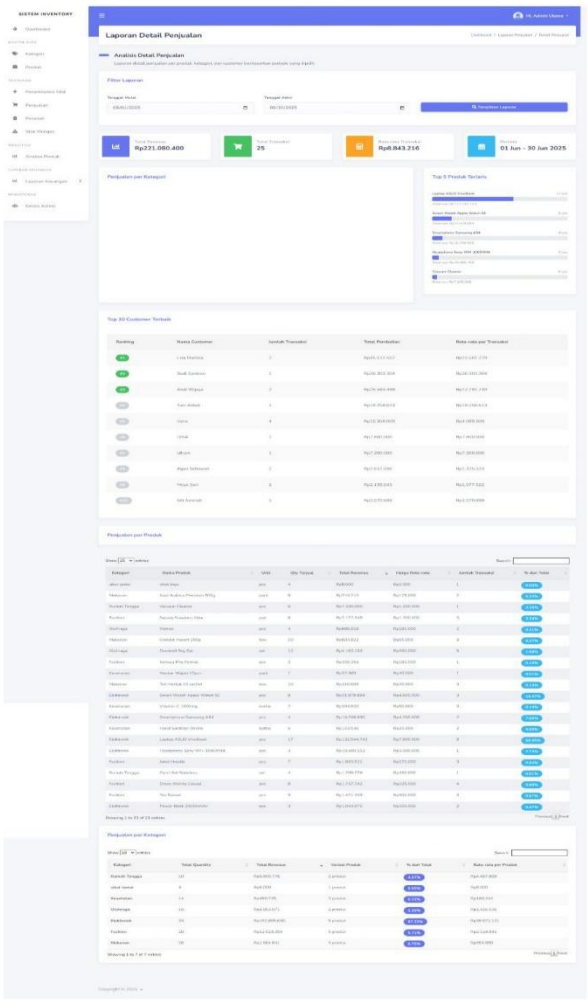
Gambar 3. 39 halaman laporan hpp

c. Laporan Harga Pokok Penjualan (COGS) menyajikan informasi penting terkait biaya langsung yang dikeluarkan untuk menghasilkan produk yang terjual. Dalam laporan ini, ditampilkan data COGS per kategori yang menunjukkan total biaya berdasarkan masing-masing kategori produk, serta COGS per produk yang merinci biaya untuk setiap produk secara individual. Selain itu, laporan ini juga mencantumkan total kuantitas produk yang terjual (volume penjualan) serta harga beli atau purchase price dari masing-masing produk. Informasi ini sangat berguna untuk analisis efisiensi biaya dan margin keuntungan yang terlihat pada gambar 3.39



Gambar 3. 40 halaman laporan laba

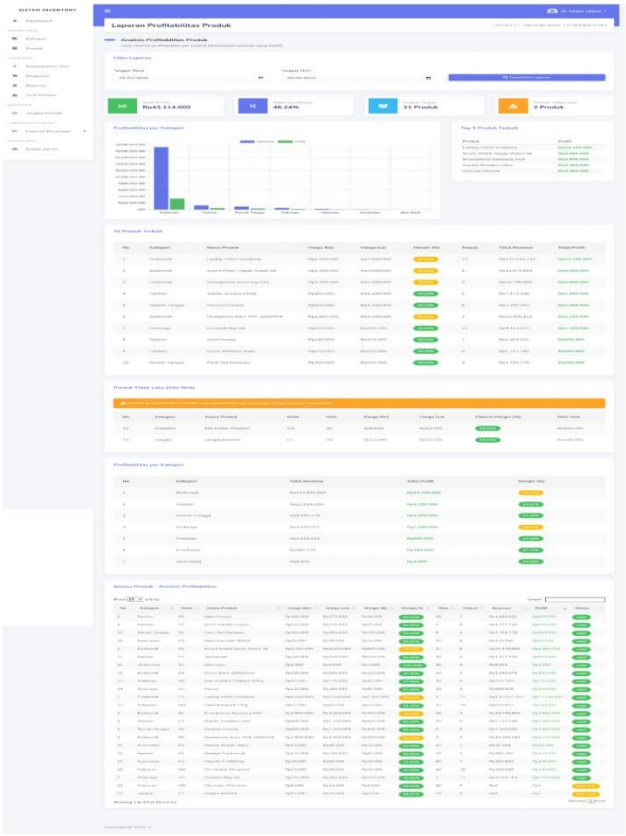
d. Laporan Gross Profit menyajikan data profitabilitas setiap produk secara rinci. Informasi yang ditampilkan meliputi kategori dan nama produk, perbandingan antara harga beli dan harga jual, total jumlah produk yang terjual, serta total pendapatan (revenue) yang dihasilkan. Selain itu, laporan ini juga menghitung nilai Gross Profit, yaitu selisih antara pendapatan dan harga pokok penjualan, serta persentase margin keuntungan. Laporan ini sangat berguna untuk menilai seberapa besar keuntungan kotor yang dihasilkan oleh masing-masing produk dan membantu dalam pengambilan keputusan bisnis yang lebih strategis yang terlihat pada gambar 3.40



Gambar 3. 41 laporan detail penjualan

e. Laporan Detail Penjualan memberikan gambaran komprehensif mengenai performa penjualan berdasarkan berbagai aspek. Pada bagian ***Penjualan per Produk***, ditampilkan informasi seperti kategori dan nama produk, jumlah unit yang terjual, total revenue yang dihasilkan, rata-rata harga jual per unit, serta jumlah transaksi yang terkait dengan produk tersebut. Selanjutnya, pada ***Penjualan per Kategori***, laporan menyajikan total kuantitas produk yang terjual dalam setiap kategori, total pendapatan per kategori, serta variasi jumlah produk yang termasuk dalam kategori

tersebut.Selain itu, terdapat bagian ***Top Customer*** yang menampilkan nama pelanggan terbaik berdasarkan jumlah transaksi dan total nilai pembelian mereka, berguna untuk mengidentifikasi pelanggan setia dan bernilai tinggi. Terakhir, pada bagian ***Ringkasan***, ditampilkan data agregat seperti total jumlah transaksi, total pendapatan (revenue), dan rata-rata nilai transaksi, yang membantu memberikan overview cepat terhadap performa penjualan secara keseluruhan yang terlihat pada gambar 3.41



Gambar 3. 42 halaman laporan probabilitas produk

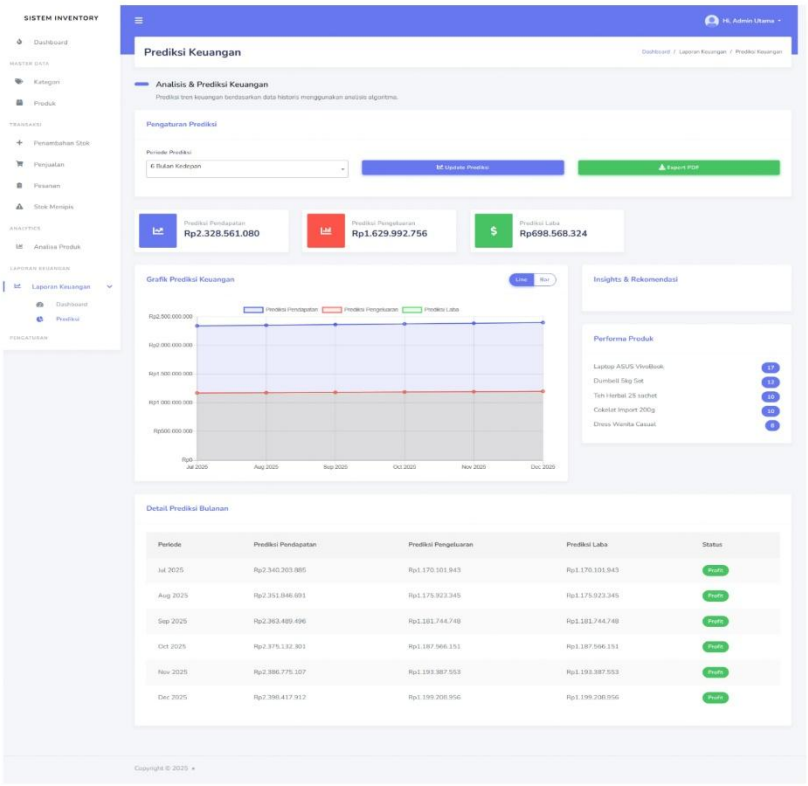
f. Laporan Profitabilitas Produk menyajikan analisis mendalam mengenai kinerja keuangan masing-masing produk berdasarkan kontribusi terhadap keuntungan. Pada bagian *Detail per Produk*, informasi yang ditampilkan meliputi margin keuntungan dalam bentuk jumlah (*Margin Amount*) dan persentase, total unit yang terjual beserta revenue yang dihasilkan, total profit yang diperoleh dari produk tersebut, serta jumlah stok saat ini.

   Di bagian *Best Performers*, sistem akan menampilkan 10 produk dengan profit tertinggi, memberikan insight terhadap produk-produk yang paling menguntungkan. Sebaliknya, bagian *Worst Performers* memuat daftar produk yang memiliki stok tersedia namun tidak mengalami penjualan, menunjukkan potensi masalah dalam permintaan atau strategi pemasaran

Laporan ini juga mencakup *Profitabilitas per Kategori* yang memperlihatkan total revenue dan profit berdasarkan kategori produk, serta margin persentase masing-masing kategori, berguna untuk menilai efektivitas tiap lini produk.

Sebagai penutup, terdapat bagian *Strategi Berdasarkan Data* yang memberikan rekomendasi langsung berdasarkan hasil analisis: produk dengan performa terbaik sebaiknya ditingkatkan stok dan promosinya, produk yang tidak laku perlu dievaluasi harga atau bahkan dihentikan, serta produk dengan margin tinggi perlu menjadi fokus utama dalam strategi penjualan yang terlihat pada gambar 3.42

3.2.9 menu prediksi keuanga



Gambar 3. 43 halaman prediksi keuangan

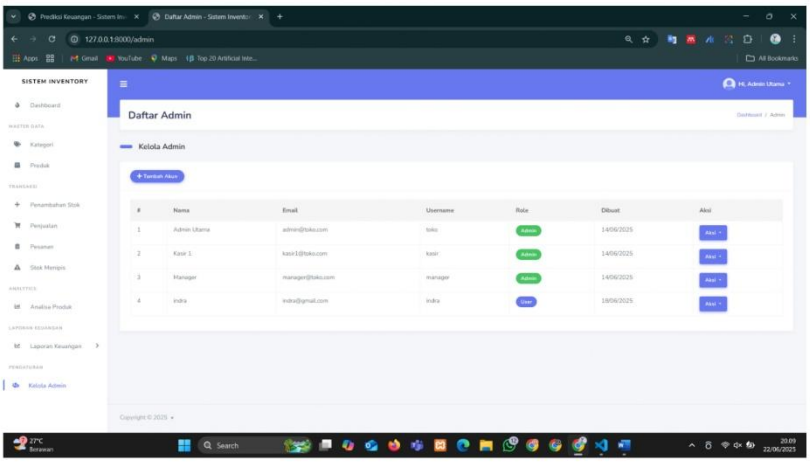
a. Modul prediksi keuangan dirancang untuk memberikan gambaran proyeksi kondisi keuangan bisnis di masa mendatang secara cepat dan informatif melalui dashboard utama. Di dalamnya terdapat dua komponen utama, yaitu *Panel Kontrol* dan *Ringkasan Prediksi*.

   Pada *Panel Kontrol*, pengguna dapat memilih rentang waktu prediksi melalui dropdown periode, dengan pilihan 3, 6, 12, atau 24 bulan ke depan, sesuai kebutuhan analisis. Setelah memilih periode, pengguna cukup menekan tombol "Update Prediksi" untuk memperbarui dan menampilkan data prediksi terbaru berdasarkan histori keuangan yang tersedia.

   Sementara itu, bagian *Ringkasan Prediksi* menyajikan tiga metrik utama yang mencerminkan kondisi keuangan masa depan, yaitu: *Total Prediksi Pendapatan* yang

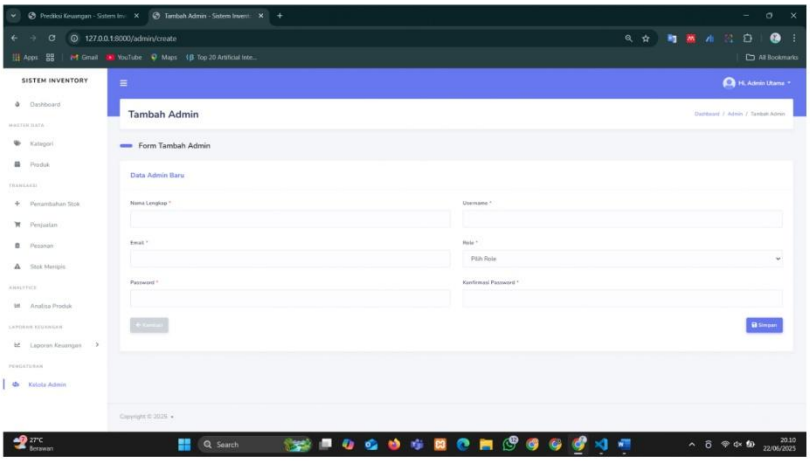
menunjukkan estimasi revenue, *Total Prediksi Pengeluaran* yang memperkirakan total biaya atau expenses, serta *Total Prediksi Keuntungan* yang merupakan hasil pengurangan antara pendapatan dan pengeluaran (profit = revenue - expenses). Ketiga metrik ini disajikan dalam format yang mudah dipahami untuk mendukung pengambilan keputusan jangka menengah hingga Panjang yang terlihat pada gambar 3.43

3.2.9 menu Kelola akun

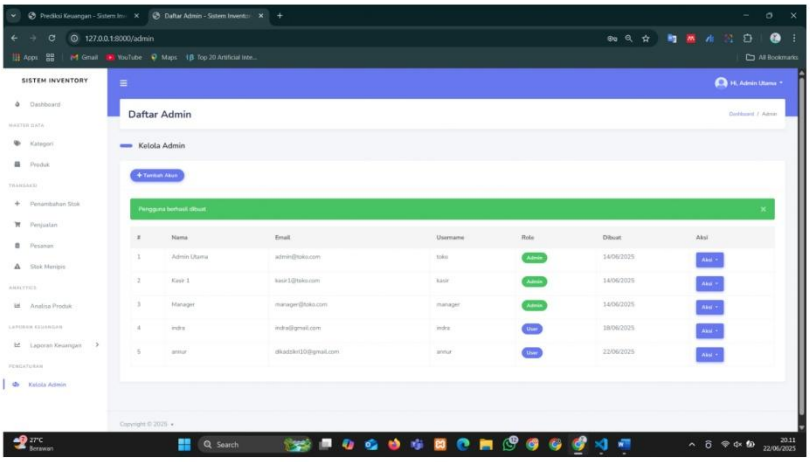


Gambar 3. 44 halamn Kelola akun

a. Untuk membuka halaman pengelolaan akun admin, silakan klik menu "Kelola Admin". Pada halaman tersebut, Anda dapat menambahkan data akun baru dengan cara mengklik tombol "Tambah Akun", kemudian isi form yang tersedia dengan informasi yang diperlukan. Setelah semua data terisi dengan benar, klik tombol "Simpan" untuk menambahkan akun admin ke dalam sistem yang terlihat pada gambar 3.44 sampai 3.46

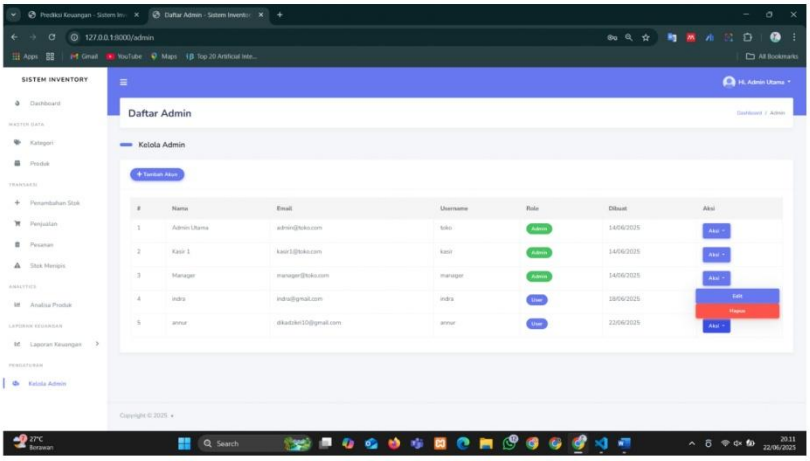


Gambar 3. 45 halaman tambah akun

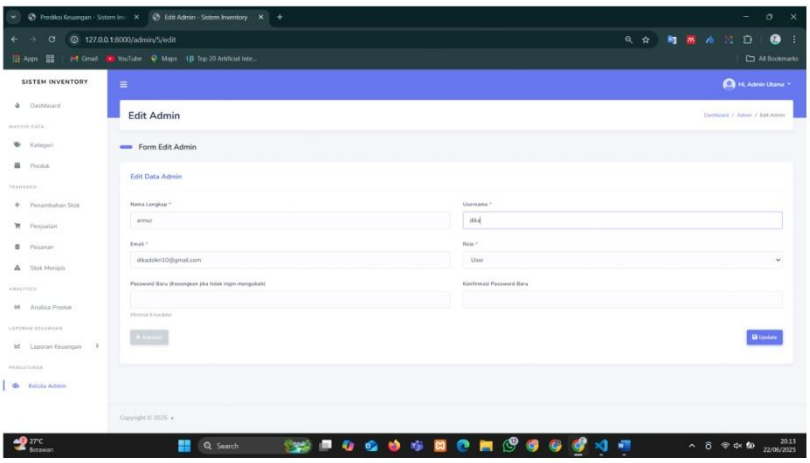


Gambar 3. 46 notifikasi penambahan data

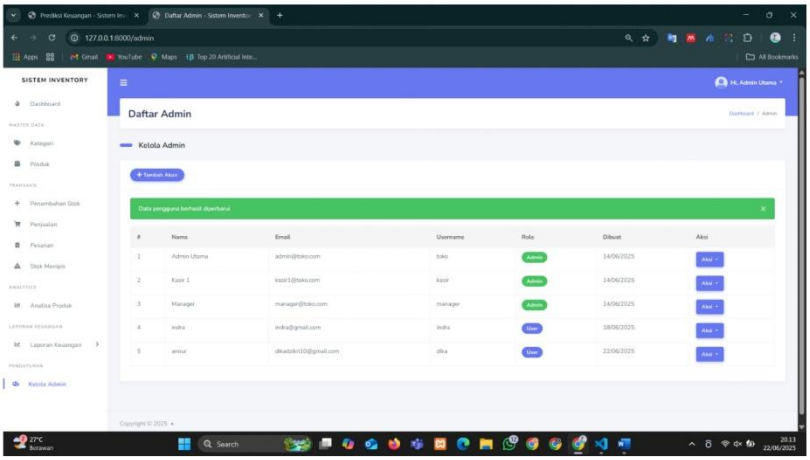
b. Untuk melakukan edit data bisa menekan menu edit dan mengubah data yang ingin di ubah dan klik update dan tersedia menu untuk menghapus data akun yang terlihat pada gambar 3.47 sampai 3. 49



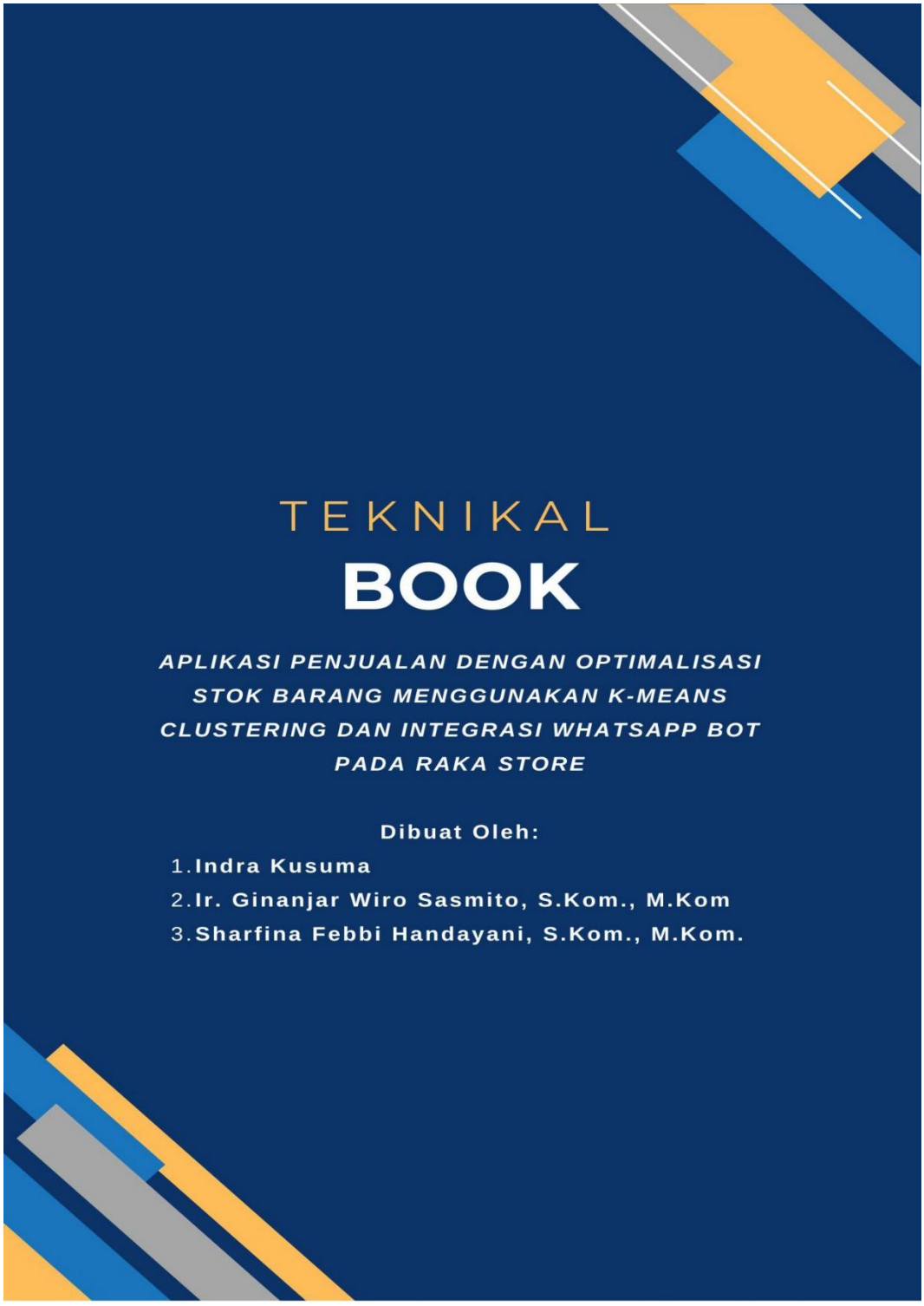
Gambar 3. 47 pilihan tombol aksi



Gambar 3. 48 halaman edit akun



Gambar 3. 49 notifikasi penambahan data


TEKNIKAL
BOOK
APLIKASI PENJUALAN DENGAN OPTIMALISASI STOK BARANG MENGGUNAKAN K-MEANS CLUSTERING DAN INTEGRASI WHATSAPP BOT PADA RAKA STORE
Dibuat Oleh:
1.Indra Kusuma
2.Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom
3.Sharfina Febbi Handayani, S.Kom., M.Kom.

Profil

Aplikasi Penjualan Raka Store merupakan sistem penjualan berbasis web yang dilengkapi dengan fitur optimasi stok barang menggunakan metode K-Means Clustering dan pengiriman notifikasi otomatis melalui WhatsApp Bot. Sistem ini dirancang untuk membantu pelaku UMKM, khususnya Raka Store, dalam memantau transaksi penjualan, mengelola stok, dan memberikan pengingat stok secara otomatis dan cerdas.

Latar Belakang

Dalam dunia usaha, pengelolaan stok barang merupakan bagian krusial dari kegiatan operasional. Stok yang tidak terkontrol dapat menyebabkan overstock (penumpukan barang) atau stockout (kehabisan barang), yang berdampak pada kerugian finansial dan menurunkan kepuasan pelanggan. Banyak UMKM mengalami kendala dalam memprediksi barang mana yang harus diutamakan pengadaannya.

Raka Store, sebagai contoh usaha kecil menengah, membutuhkan sistem penjualan yang tidak hanya mencatat transaksi, tetapi juga mampu mengoptimalkan manajemen stok. Dengan memanfaatkan metode K-Means Clustering, sistem dapat melakukan klasterisasi produk berdasarkan frekuensi penjualan, sehingga dapat mengelompokkan produk ke dalam kategori *fast moving*, *medium moving*, dan *slow moving*.

Selain itu, proses pemantauan stok seringkali memakan waktu dan rawan terabaikan. Oleh karena itu, sistem ini juga diintegrasikan dengan WhatsApp Bot, yang secara otomatis akan mengirimkan notifikasi kepada pemilik toko jika stok produk mendekati ambang batas minimum. WhatsApp dipilih karena merupakan media komunikasi yang paling sering digunakan oleh pelaku UMKM.

Dengan kombinasi data mining (K-Means) dan automasi pesan melalui WhatsApp API, aplikasi ini dapat membantu pemilik toko mengambil keputusan yang lebih baik, cepat, dan efisien dalam mengelola stok barang.

Manfaat

Adapun manfaat yang diperoleh dari penelitian ada tiga, diantaranya sebagai berikut:

1. Manfaat untuk penulis:
   a. Menerapkan ilmu yang telah dipelajari dalam bidang data mining, web programming, dan API integrasi
   b. Menambah portofolio dalam pengembangan aplikasi berbasis analisis data dan otomasi komunikasi.
2. Manfaat untuk Pengguna (UMKM / Toko) :
   a. Memudahkan pemantauan transaksi dan stok secara digital.
   b. Memberikan rekomendasi stok berdasarkan klasifikasi performa penjualan.
   c. Mengirimkan notifikasi otomatis melalui WhatsApp untuk mempercepat pengambilan keputusan.

Spesifikasi Teknis

Spesifikasi Teknis meliputi:

1. Modul Pengguna
2. Source Code

Berikut uraian spesifikasi untuk pembangunan game:

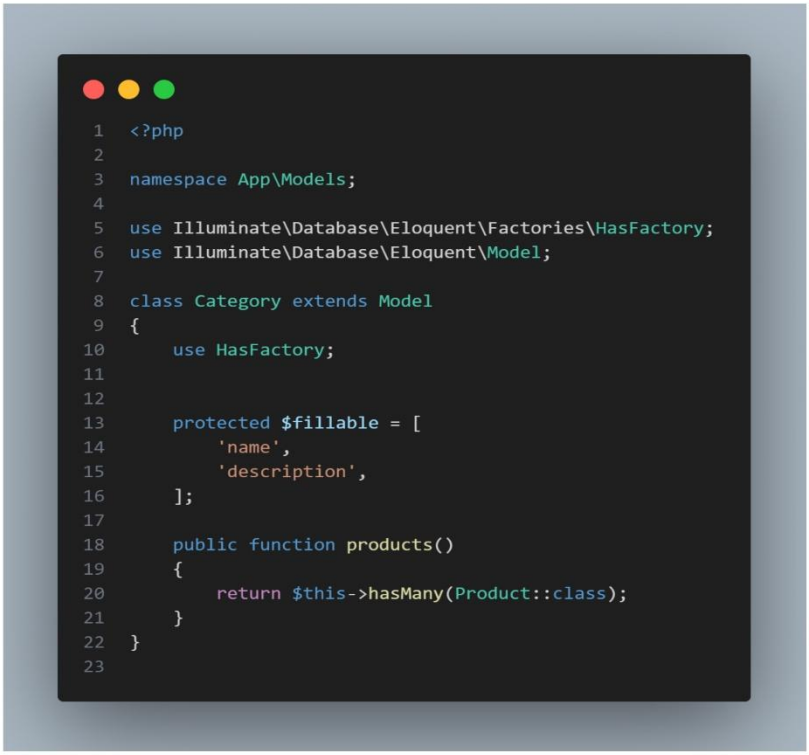
1. Backend: Laravel (PHP Framework)
2. Frontend: Blade Template / Bootstrap
3. Database: MySQL
4. Data Mining: Python (K-Means Clustering)
5. Integrasi WA Bot: Fonnte API / WhatsApp Gateway

Berikut uraian spesifikasi modul:

1. Modul Pengguna

a. Menu Dashboard
b. Menu kategori
c. Menu produk
d. Menu stok
e. Menu pesanan
f. Menu penjualan
g. Menu Analisa produk
h. Menu laporam keuangan
i. Menu prediksi keuangan
j. Menu Kelola akun

SOURCE CODE



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    use HasFactory;


    protected $fillable = [
        'name',
        'description',
    ];

    public function products()
    {
        return $this->hasMany(Product::class);
    }
}

```

protected $fillable = ['name', 'description']; Bagian ini menentukan kolom-kolom yang diizinkan untuk di-*mass assign*, yaitu pengisian data secara massal melalui method seperti create() atau update(). Dengan hanya menyertakan name dan description, kita mencegah field lain diisi sembarangan, yang sangat penting untuk melindungi aplikasi dari *mass assignment vulnerability*.

public function products() Fungsi ini mendefinisikan relasi antara model Category dan Product dengan tipe relasi **One to Many**. Artinya, satu kategori dapat memiliki banyak produk. hasMany(Product::class) membuat Eloquent otomatis tahu bahwa saat kita memanggil $category->products, yang dikembalikan adalah daftar produk dengan category_id sesuai.

```php
public function index()
    {
        $categories = Category::orderBy('name')->get();

        return view('pages.category.page-category', compact('categories'));
    }
```

Fungsi ini digunakan untuk menampilkan daftar semua kategori. Data diambil dari database, diurutkan berdasarkan nama (ascending), lalu dikirim ke tampilan page-category menggunakan helper compact.

```php
public function create()
    {
        return view('pages.category.tambah-category');
    }
```

Method create() hanya mengembalikan view yang berisi form untuk menambahkan kategori baru. Tidak ada logika pengolahan data, hanya penyajian UI.

```php
public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255|unique:categories',
            'description' => 'nullable|string',
        ]);

        Category::create($request->all());

        return redirect()->route('categories.index')
            ->with('success', 'Kategori berhasil ditambahkan!');
    }
```

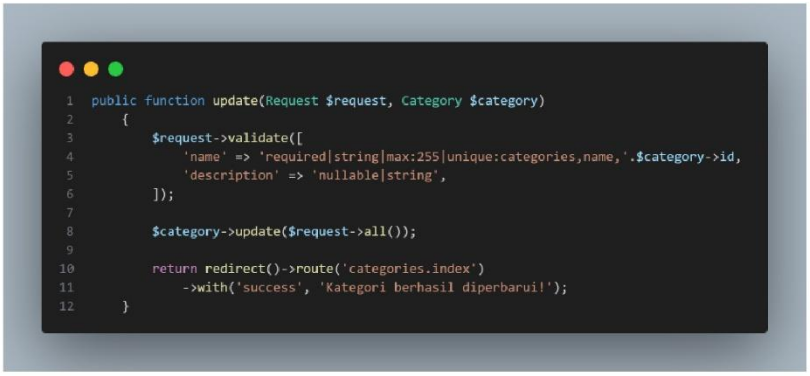
Fungsi ini menangani penyimpanan data baru dari form tambah kategori. Input dari pengguna divalidasi terlebih dahulu: name harus unik dan wajib diisi, sedangkan description opsional. Jika validasi lolos, data disimpan ke database menggunakan Category::create(), lalu pengguna diarahkan kembali ke halaman index dengan notifikasi sukses.

```php
public function show(Category $category)
    {
        return view('pages.category.detail', compact('category'));
    }

```

Fungsi show() menampilkan detail dari satu kategori. Berkat fitur route model binding Laravel, parameter $category langsung diisi oleh sistem berdasarkan ID dari URL.

```php
public function edit(Category $category)
    {
        return view('pages.category.edit-category', compact('category'));
    }
```

Fungsi ini mengembalikan tampilan form edit untuk kategori tertentu. Data kategori yang akan diedit disiapkan dalam $category dan dikirim ke view.



```php
public function update(Request $request, Category $category)
    {
        $request->validate([
            'name' => 'required|string|max:255|unique:categories,name,'.$category->id,
            'description' => 'nullable|string',
        ]);

        $category->update($request->all());

        return redirect()->route('categories.index')
            ->with('success', 'Kategori berhasil diperbarui!');
    }
```

Fungsi update() menangani logika pembaruan data kategori. Validasi memastikan nama kategori tetap unik, kecuali jika tidak berubah (dikecualikan dengan ID saat ini). Jika valid, data diperbarui dan pengguna diarahkan kembali ke halaman daftar kategori.

```php
public function destroy(Category $category)
    {
        // Cek apakah kategori digunakan di produk
        if ($category->products()->count() > 0) {
            return redirect()->route('categories.index')
                ->with('error', 'Kategori tidak dapat dihapus karena masih digunakan oleh produk!');
        }

        $category->delete();

        return redirect()->route('categories.index')
            ->with('success', 'Kategori berhasil dihapus!');
    }
```

Fungsi destroy()

Sebelum menghapus data kategori, fungsi ini mengecek apakah masih ada produk yang berelasi dengan kategori tersebut. Jika masih ada, proses penghapusan dibatalkan dan ditampilkan pesan error. Jika tidak ada produk, kategori dihapus dan ditampilkan pesan sukses. Ini menjaga agar integritas data tetap aman.

MODEL: PRODUCT.PHP

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;

    protected $fillable = [
        'code',
        'name',
        'category_id',
        'stock',
        'min_stock',
        'purchase_price',
        'selling_price',
        'unit',
        'image',
    ];

    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    public function stockInDetails()
    {
        return $this->hasMany(StockInDetail::class);
    }

    public function saleDetails()
    {
        return $this->hasMany(SaleDetail::class);
    }

    public function orderDetails()
    {
        return $this->hasMany(OrderDetail::class);
    }

    // public function stockAdjustments()
    // {
    //     return $this->hasMany(StockAdjustment::class);
    // }

    // Status stok (normal, rendah, habis)
    public function getStockStatusAttribute()
    {
        if ($this->stock <= 0) {
            return 'out_of_stock';
        } elseif ($this->stock <= $this->min_stock) {
            return 'low_stock';
        } else {
            return 'in_stock';
        }
    }
}

```

- protected $fillable
  Properti ini menentukan atribut mana saja yang bisa di-*mass assign* melalui create() atau update(). Tujuannya untuk keamanan, agar hanya field tertentu yang bisa diisi langsung dari input user.
- public function category()
  Fungsi ini menyatakan bahwa sebuah produk dimiliki oleh satu kategori. Relasi belongsTo digunakan untuk relasi satu ke satu dari sisi produk ke kategori.
- Relasi: stockInDetails(), saleDetails(), orderDetails()
  Relasi hasMany ini menyatakan bahwa satu produk dapat memiliki banyak entri detail stok masuk, penjualan, maupun pesanan. Relasi-relasi ini sangat penting dalam pelacakan riwayat dan laporan transaksi.
- getStockStatusAttribute()
  Fungsi ini membuat atribut virtual stock_status yang otomatis menentukan status stok produk: kosong, rendah, atau aman. Ini berguna untuk sistem peringatan stok.

PRODUCT CONTROLLER.PHP

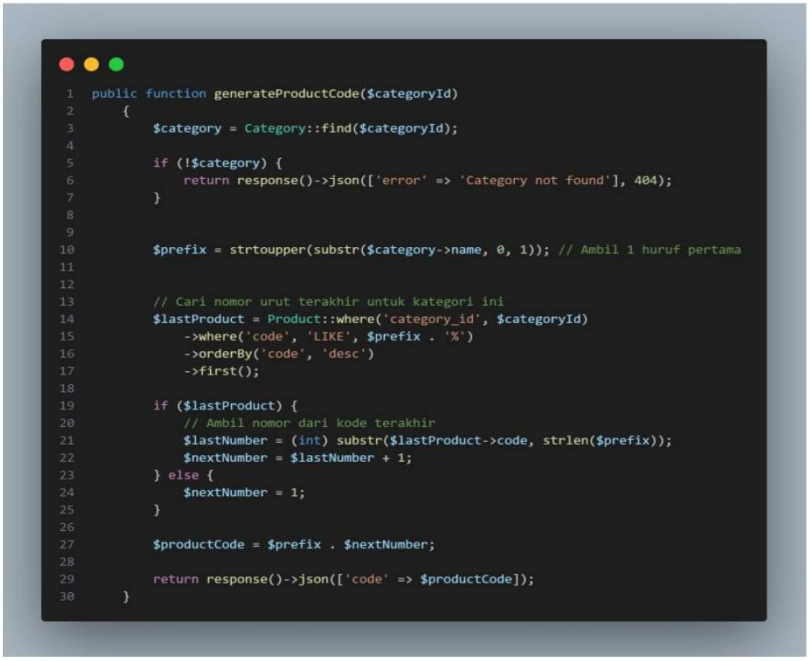
```
public function index()
    {
        $products = Product::with('category')->get();
        return view('pages.produk.index', compact('products'));
    }
```

- Fungsi index()
  Menampilkan semua produk beserta relasi kategori-nya. Data dikirim ke view pages.produk.index.

```
public function create()
    {
        $categories = Category::all();
        return view('pages.produk.tambah-pruduk', compact('categories'));
    }

```

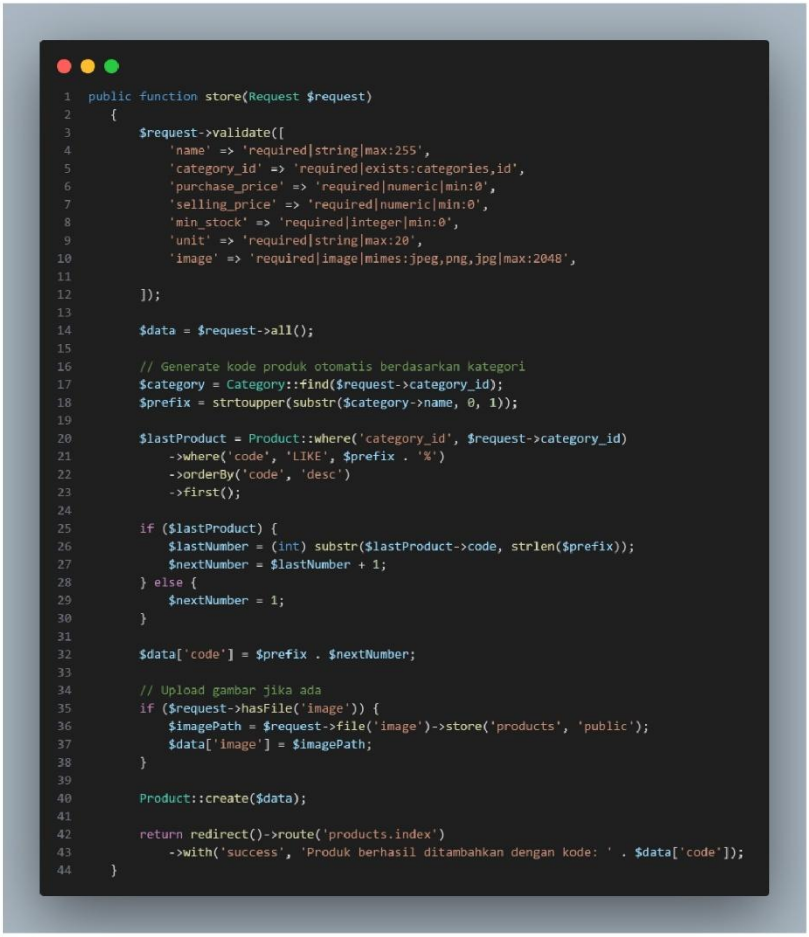
- Fungsi create()

  Mengambil semua kategori sebagai pilihan dalam form tambah produk.



```php
public function generateProductCode($categoryId)
    {
        $category = Category::find($categoryId);

        if (!$category) {
            return response()->json(['error' => 'Category not found'], 404);
        }


        $prefix = strtoupper(substr($category->name, 0, 1)); // Ambil 1 huruf pertama


        // Cari nomor urut terakhir untuk kategori ini
        $lastProduct = Product::where('category_id', $categoryId)
            ->where('code', 'LIKE', $prefix . '%')
            ->orderBy('code', 'desc')
            ->first();

        if ($lastProduct) {
            // Ambil nomor dari kode terakhir
            $lastNumber = (int) substr($lastProduct->code, strlen($prefix));
            $nextNumber = $lastNumber + 1;
        } else {
            $nextNumber = 1;
        }

        $productCode = $prefix . $nextNumber;

        return response()->json(['code' => $productCode]);
    }
```

- Fungsi generateProductCode($categoryId)
  Kode produk dibuat otomatis berdasarkan huruf awal kategori. Nomor urut terakhir dicek dan ditambahkan 1.



```php
public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'category_id' => 'required|exists:categories,id',
            'purchase_price' => 'required|numeric|min:0',
            'selling_price' => 'required|numeric|min:0',
            'min_stock' => 'required|integer|min:0',
            'unit' => 'required|string|max:20',
            'image' => 'required|image|mimes:jpeg,png,jpg|max:2048',

        ]);

        $data = $request->all();

        // Generate kode produk otomatis berdasarkan kategori
        $category = Category::find($request->category_id);
        $prefix = strtoupper(substr($category->name, 0, 1));

        $lastProduct = Product::where('category_id', $request->category_id)
            ->where('code', 'LIKE', $prefix . '%')
            ->orderBy('code', 'desc')
            ->first();

        if ($lastProduct) {
            $lastNumber = (int) substr($lastProduct->code, strlen($prefix));
            $nextNumber = $lastNumber + 1;
        } else {
            $nextNumber = 1;
        }

        $data['code'] = $prefix . $nextNumber;

        // Upload gambar jika ada
        if ($request->hasFile('image')) {
            $imagePath = $request->file('image')->store('products', 'public');
            $data['image'] = $imagePath;
        }

        Product::create($data);

        return redirect()->route('products.index')
            ->with('success', 'Produk berhasil ditambahkan dengan kode: ' . $data['code']);
    }
```

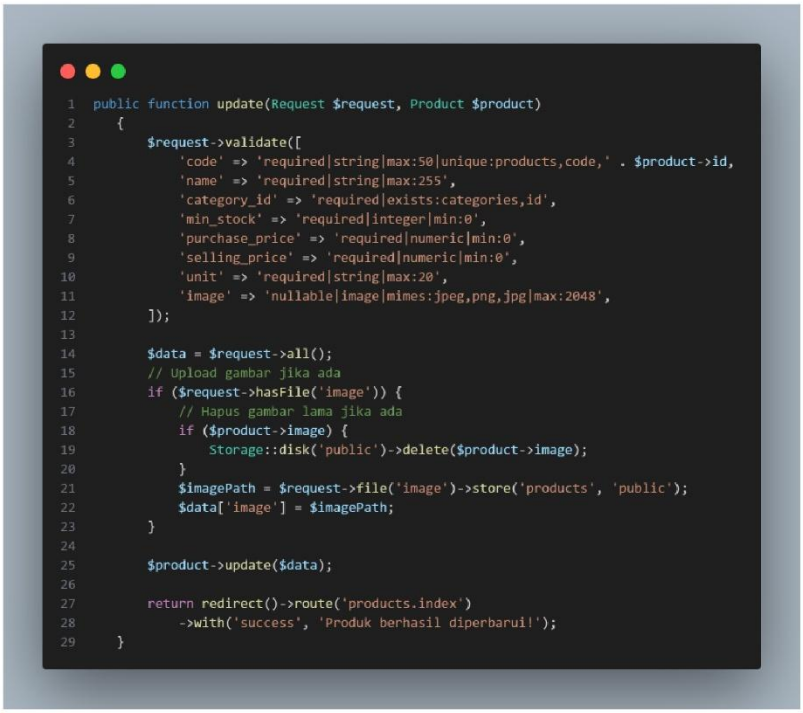
- Fungsi store()
  Setelah input data produk diterima melalui form, sistem akan melakukan proses validasi untuk memastikan bahwa semua field yang diperlukan telah diisi dengan benar. Validasi mencakup pengecekan terhadap nama produk, harga beli dan jual, stok minimum, unit, kategori, serta file gambar yang diunggah. Jika semua data dinyatakan valid, maka sistem akan secara otomatis menghasilkan kode produk berdasarkan huruf awal nama kategori dan nomor urut terakhir dari produk dalam kategori tersebut. Selanjutnya, apabila terdapat file gambar, sistem akan mengunggah file tersebut ke direktori penyimpanan storage/app/public/products, dan path gambar akan disimpan dalam database. Setelah itu, seluruh data produk termasuk kode, informasi harga, dan gambar—akan disimpan ke tabel products menggunakan metode create().

```php
public function show(Product $product)
    {
        return view('pages.produk.detail-produk', compact('product'));
    }
```

- Fungsi show(Product $product)
- Menampilkan detail produk berdasarkan ID.

```php
public function edit(Product $product)
    {
        $categories = Category::all();
        return view('pages.produk.edit-produk', compact('product', 'categories'));
    }
```

- Fungsi edit(Product $product)
  Mengembalikan form edit dengan data produk dan kategori yang tersedia.



```php
public function update(Request $request, Product $product)
    {
        $request->validate([
            'code' => 'required|string|max:50|unique:products,code,' . $product->id,
            'name' => 'required|string|max:255',
            'category_id' => 'required|exists:categories,id',
            'min_stock' => 'required|integer|min:0',
            'purchase_price' => 'required|numeric|min:0',
            'selling_price' => 'required|numeric|min:0',
            'unit' => 'required|string|max:20',
            'image' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
        ]);

        $data = $request->all();
        // Upload gambar jika ada
        if ($request->hasFile('image')) {
            // Hapus gambar lama jika ada
            if ($product->image) {
                Storage::disk('public')->delete($product->image);
            }
            $imagePath = $request->file('image')->store('products', 'public');
            $data['image'] = $imagePath;
        }

        $product->update($data);

        return redirect()->route('products.index')
            ->with('success', 'Produk berhasil diperbarui!');
    }
```

- Fungsi update()
  Saat proses pembaruan data produk, sistem akan terlebih dahulu memeriksa apakah ada file gambar baru yang diunggah oleh pengguna. Jika ada, maka gambar lama yang tersimpan sebelumnya akan dihapus dari penyimpanan. Selanjutnya, gambar baru akan disimpan ke dalam folder yang telah ditentukan. Setelah semua data siap, sistem akan memperbarui informasi produk di database sesuai dengan input terbaru.

```php
public function destroy($id)
    {
        try {
            // Cari produk berdasarkan ID
            $product = Product::findOrFail($id);

            // Cek apakah produk sedang digunakan dalam transaksi
            $isUsedInTransactions = $this->checkProductUsage($product->id);

            if ($isUsedInTransactions) {
                if (request()->ajax()) {
                    return response()->json([
                        'success' => false,
                        'message' => 'Tidak dapat menghapus produk karena sudah digunakan dalam transaksi penjualan!'
                    ], 400);
                }

                return redirect()->route('products.index')
                    ->with('error', 'Tidak dapat menghapus produk karena sudah digunakan dalam transaksi penjualan!');
            }

            // Simpan nama produk untuk response
            $productName = $product->name;

            // Hapus gambar jika ada
            if ($product->image && Storage::disk('public')->exists($product->image)) {
                Storage::disk('public')->delete($product->image);
            }

            // Hapus produk
            $product->delete();

            // Response untuk AJAX request
            if (request()->ajax()) {
                return response()->json([
                    'success' => true,
                    'message' => "Produk '{$productName}' berhasil dihapus!"
                ]);
            }

            // Response untuk normal request
            return redirect()->route('products.index')
                ->with('success', "Produk '{$productName}' berhasil dihapus!");
        } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {
            if (request()->ajax()) {
                return response()->json([
                    'success' => false,
                    'message' => 'Produk tidak ditemukan!'
                ], 404);
            }

            return redirect()->route('products.index')
                ->with('error', 'Produk tidak ditemukan!');
        } catch (\Exception $e) {
            if (request()->ajax()) {
                return response()->json([
                    'success' => false,
                    'message' => 'Terjadi kesalahan: ' . $e->getMessage()
                ], 500);
            }

            return redirect()->route('products.index')
                ->with('error', 'Terjadi kesalahan: ' . $e->getMessage());
        }
    }

```

- Function destroy($id)
  Menghapus produk dengan pengecekan: jika produk masih digunakan dalam tabel transaksi (penjualan, pesanan, dsb), maka tidak bisa dihapus. Juga menghapus file gambar dari penyimpanan.

```php
public function lowStock()
    {
        $products = Product::whereRaw('stock <= min_stock')->with('category')->get();

        // $this->sendMessage();
        return view('pages.stok.low-stock', compact('products'));
    }
```

- Fungsi lowStock()
  Mengambil semua produk yang stoknya lebih rendah dari minimum. Data ini ditampilkan dalam halaman low-stock, bisa digunakan untuk keperluan monitoring dan alert.

MODEL ORDER

Model Order merepresentasikan entitas pesanan dalam sistem. Ini menyimpan informasi penting pelanggan, rincian pembayaran, serta mengelola hubungan terhadap detail produk, pengguna (user), dan penjualan (sale).

```php
class Order extends Model
{
    use HasFactory;
    protected $fillable = [
        'order_number',
        'customer_name',
        'customer_phone',
        'customer_address',
        'order_date',
        'expected_delivery',
        'status',
        'total_amount',
        'discount',
        'tax',
        'final_amount',
        'down_payment',
        'notes',
        'user_id',
        'Product_id', // Tambahkan kolom ini
    ];
```

Properti ini menentukan kolom-kolom yang diizinkan untuk di-*mass assign*. Artinya, hanya kolom-kolom ini yang dapat diisi melalui method seperti create() atau update(). Ini penting untuk menjaga keamanan dan memastikan data yang disimpan adalah data yang valid. Field Product_id sebaiknya diubah menjadi lowercase (product_id) untuk konsistensi penulisan.

```php
protected $dates = [
        'order_date',
        'expected_delivery',
    ];
    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function orderDetails()
    {
        return $this->hasMany(OrderDetail::class);
    }

    public function products()
    {
        return $this->belongsToMany(Product::class, 'order_details')
            ->withPivot('quantity', 'selling_price', 'discount', 'subtotal');
    }

    public function sale()
    {
        return $this->hasOne(Sale::class);
    }

```

- protected $dates

  Kolom-kolom ini akan otomatis diperlakukan sebagai objek tanggal (`Carbon`) oleh Laravel. Ini memudahkan manipulasi tanggal seperti formatting, perhitungan selisih waktu, dan sebagainya.

- public function user()

  Relasi ini menunjukkan bahwa setiap order dibuat oleh satu pengguna. Ini adalah relasi belongsTo, artinya foreign key user_id berada di tabel orders.

- public function orderDetails()

  Fungsi ini menunjukkan bahwa satu order bisa memiliki banyak detail pesanan. Ini menggunakan relasi hasMany untuk menghubungkan ke model OrderDetail.

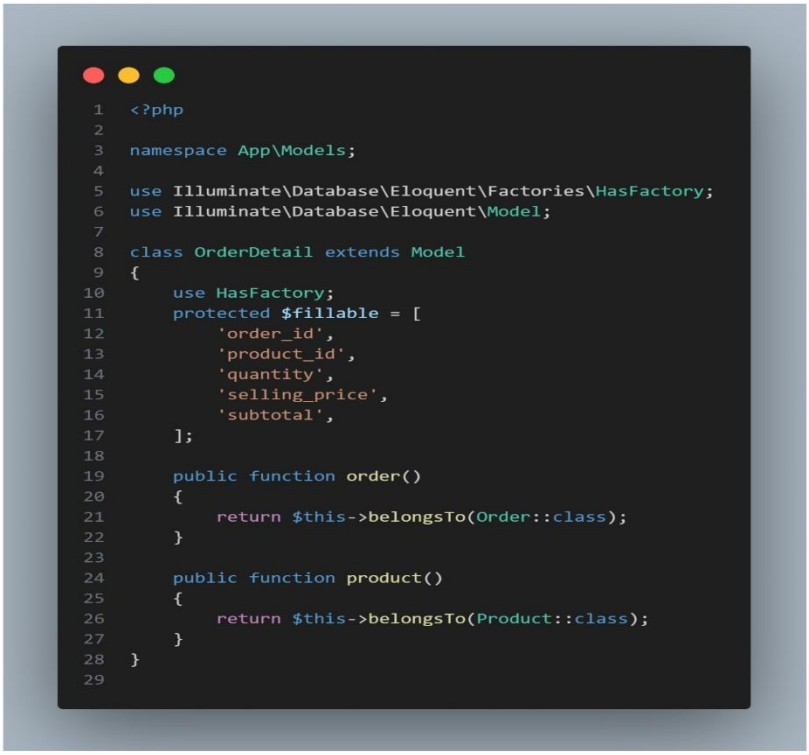
- public function products()

  Fungsi ini mendefinisikan relasi many-to-many antara order dan produk melalui tabel pivot order_details. Metode withPivot() digunakan agar data tambahan seperti kuantitas, harga jual, diskon, dan subtotal bisa diakses langsung dari relasi ini. Ini sangat berguna saat menampilkan ringkasan atau invoice pesanan.

- public function sale()

  Order bisa dikaitkan ke satu transaksi penjualan (sale). Relasi hasOne berarti setiap order hanya boleh memiliki satu entri sale, yang biasanya menyimpan informasi seperti metode pembayaran, tanggal pembayaran, dan status pelunasan.

MODEL ORDER DETAIL



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class OrderDetail extends Model
{
    use HasFactory;
    protected $fillable = [
        'order_id',
        'product_id',
        'quantity',
        'selling_price',
        'subtotal',
    ];

    public function order()
    {
        return $this->belongsTo(Order::class);
    }

    public function product()
    {
        return $this->belongsTo(Product::class);
    }
}

```

- protected $fillable
  Properti ini menentukan atribut-atribut yang dapat di-*mass assign*, yaitu atribut yang bisa langsung diisi secara otomatis saat membuat atau memperbarui model. Dalam konteks ini, field seperti order_id, product_id, quantity, dan lainnya dapat langsung diisi saat melakukan transaksi pemesanan. Ini menjaga keamanan dan konsistensi data.
- public function order()
  Relasi ini menunjukkan bahwa satu detail pesanan (OrderDetail) termasuk dalam satu entitas pemesanan utama (Order). Ini adalah relasi belongsTo karena order_id disimpan di tabel order_details.
- public function product()
  Relasi ini menghubungkan detail pesanan ke satu produk tertentu. Melalui relasi belongsTo, kamu dapat mengakses data produk dari setiap baris detail pesanan. Misalnya, $orderDetail->product->name akan mengembalikan nama produk dari detail tersebut.

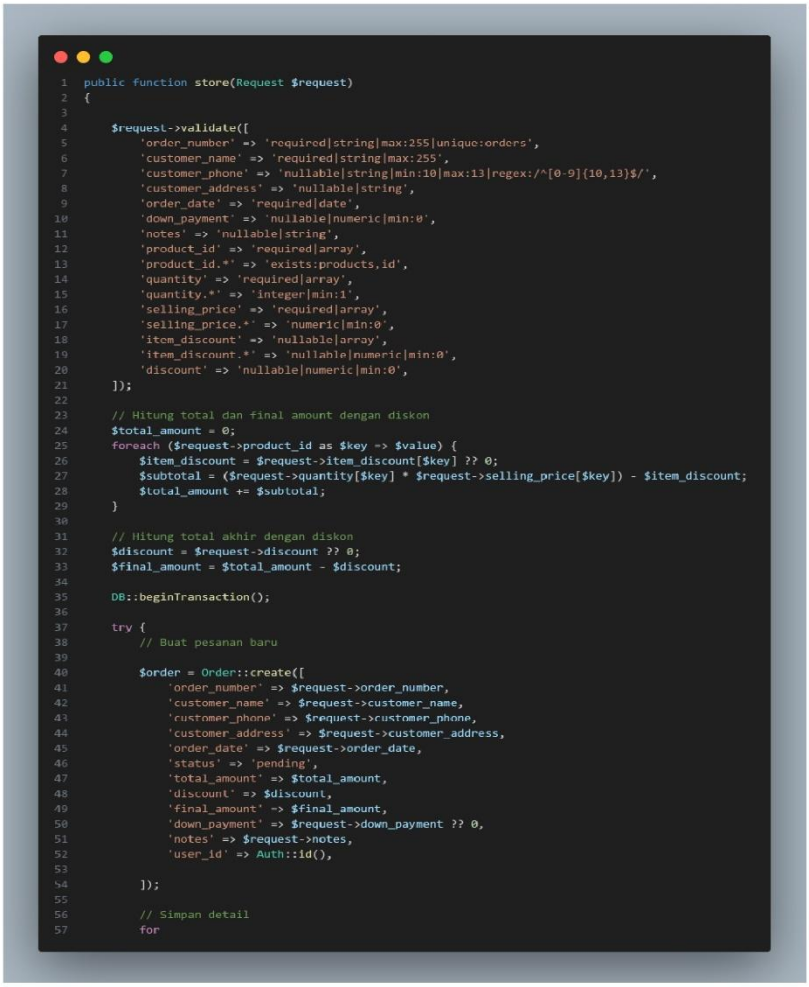
CONTROLLER ORDER

```php
public function index()
{
    try {
        $orders = Order::with('user')  // ← Pastikan nama variabel $orders
            ->orderBy('order_date', 'desc')
            ->latest()
            ->get();

        return view('pages.pesanan.index', compact('orders'));
    } catch (\Exception $e) {
        $orders = collect([]);
        return view('pages.pesanan.index', compact('orders'))
            ->with('error', 'Terjadi kesalahan saat memuat data pesanan: ' . $e->getMessage());
    }
}
```

- Fungsi ini menampilkan seluruh daftar pesanan. Data diambil dari tabel orders dan dimuat bersama relasi user, lalu diurutkan berdasarkan tanggal pesanan. Jika terjadi error saat proses pemuatan, sistem menangkap exception dan mengirim pesan error ke tampilan.

```php
public function create()
{
    // Hapus filter status karena tidak ada kolom status di tabel products
    $products = Product::all();

    // Generate nomor pesanan dengan database lock
    $order_number = DB::transaction(function () {
        $prefix = 'ORD ';
        $date = date('Ymd');

        // Gunakan row locking untuk mencegah race condition
        $lastOrder = Order::where('order_number', 'like', "{$prefix}{$date}%")
            ->lockForUpdate()
            ->orderBy('order_number', 'desc')
            ->first();

        if ($lastOrder) {
            $lastNumber = intval(substr($lastOrder ->order_number,  4));
            $newNumber = $lastNumber + 1;
        } else {
            $newNumber = 1;
        }

        return $prefix . $date . str_pad($newNumber, 4, '0', STR_PAD_LEFT);
    });

    return view('pages.pesanan.tambah-pesanan', compact('products', 'order_number'));
}
```

- Fungsi ini menampilkan form untuk membuat pesanan baru. Di dalamnya terdapat proses pembuatan nomor pesanan (order_number) secara otomatis. Nomor ini digenerate berdasarkan tanggal dan urutan terakhir pada hari itu menggunakan transaksi database dan lockForUpdate untuk mencegah konflik saat banyak pengguna membuat pesanan bersamaan.

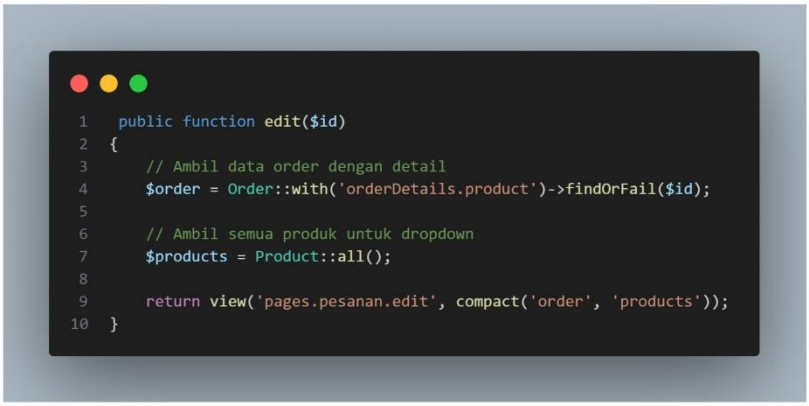


```php
public function store(Request $request)
{

    $request->validate([
        'order_number' => 'required|string|max:255|unique:orders',
        'customer_name' => 'required|string|max:255',
        'customer_phone' => 'nullable|string|min:10|max:13|regex:/^[0-9]{10,13}$/',
        'customer_address' => 'nullable|string',
        'order_date' => 'required|date',
        'down_payment' => 'nullable|numeric|min:0',
        'notes' => 'nullable|string',
        'product_id' => 'required|array',
        'product_id.*' => 'exists:products,id',
        'quantity' => 'required|array',
        'quantity.*' => 'integer|min:1',
        'selling_price' => 'required|array',
        'selling_price.*' => 'numeric|min:0',
        'item_discount' => 'nullable|array',
        'item_discount.*' => 'nullable|numeric|min:0',
        'discount' => 'nullable|numeric|min:0',
    ]);

    // Hitung total dan final amount dengan diskon
    $total_amount = 0;
    foreach ($request->product_id as $key => $value) {
        $item_discount = $request->item_discount[$key] ?? 0;
        $subtotal = ($request->quantity[$key] * $request->selling_price[$key]) - $item_discount;
        $total_amount += $subtotal;
    }

    // Hitung total akhir dengan diskon
    $discount = $request->discount ?? 0;
    $final_amount = $total_amount - $discount;

    DB::beginTransaction();

    try {
        // Buat pesanan baru

        $order = Order::create([
            'order_number' => $request->order_number,
            'customer_name' => $request->customer_name,
            'customer_phone' => $request->customer_phone,
            'customer_address' => $request->customer_address,
            'order_date' => $request->order_date,
            'status' => 'pending',
            'total_amount' => $total_amount,
            'discount' => $discount,
            'final_amount' => $final_amount,
            'down_payment' => $request->down_payment ?? 0,
            'notes' => $request->notes,
            'user_id' => Auth::id(),

        ]);

        // Simpan detail
        for
```

- Fungsi ini menyimpan data pesanan baru ke database. Data dari form divalidasi terlebih dahulu, termasuk informasi pelanggan, tanggal, dan detail produk. Sistem menghitung total dan nilai akhir transaksi, lalu menyimpan data ke tabel orders dan order_details. Semua operasi dibungkus dalam transaksi database agar jika terjadi error, proses dibatalkan (rollback).

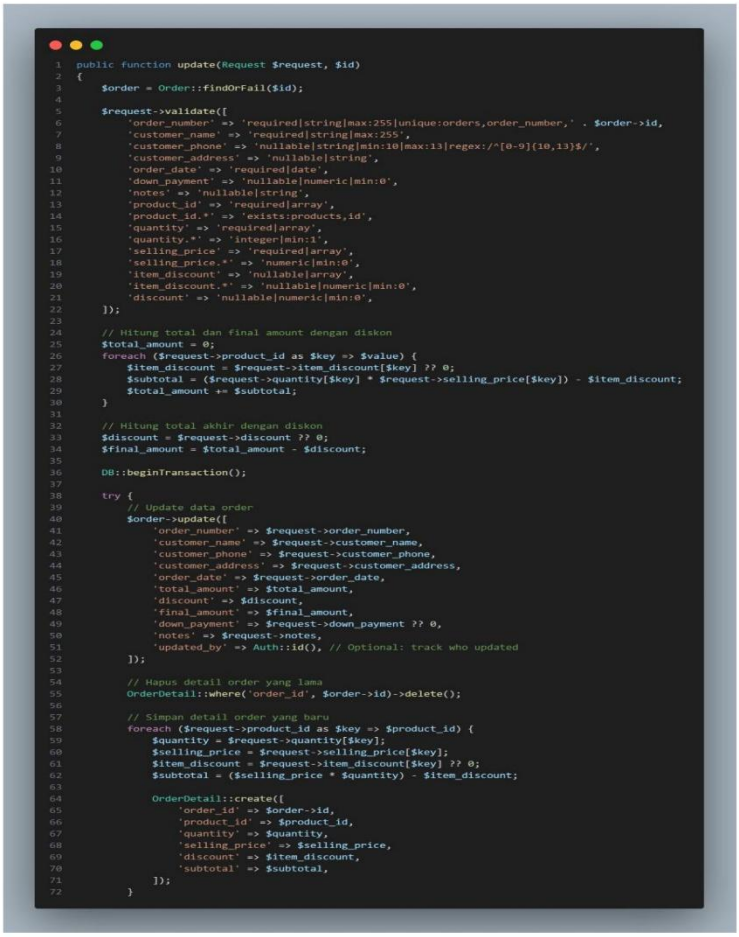
```php
public function show(Order $order)
  {
      $order->load(['user', 'orderDetails.product', 'sale']);

      return view('pages.pesanan.detail', compact('order'));
  }
```

- Menampilkan detail lengkap satu pesanan, termasuk informasi user, daftar produk yang dipesan, dan jika ada, data penjualannya.



```php
 public function edit($id)
{
    // Ambil data order dengan detail
    $order = Order::with('orderDetails.product')->findOrFail($id);

    // Ambil semua produk untuk dropdown
    $products = Product::all();

    return view('pages.pesanan.edit', compact('order', 'products'));
}
```

- Fungsi ini mengambil data pesanan tertentu beserta detail produk untuk ditampilkan dalam form edit. Juga memuat daftar semua produk untuk dipilih ulang jika diperlukan.

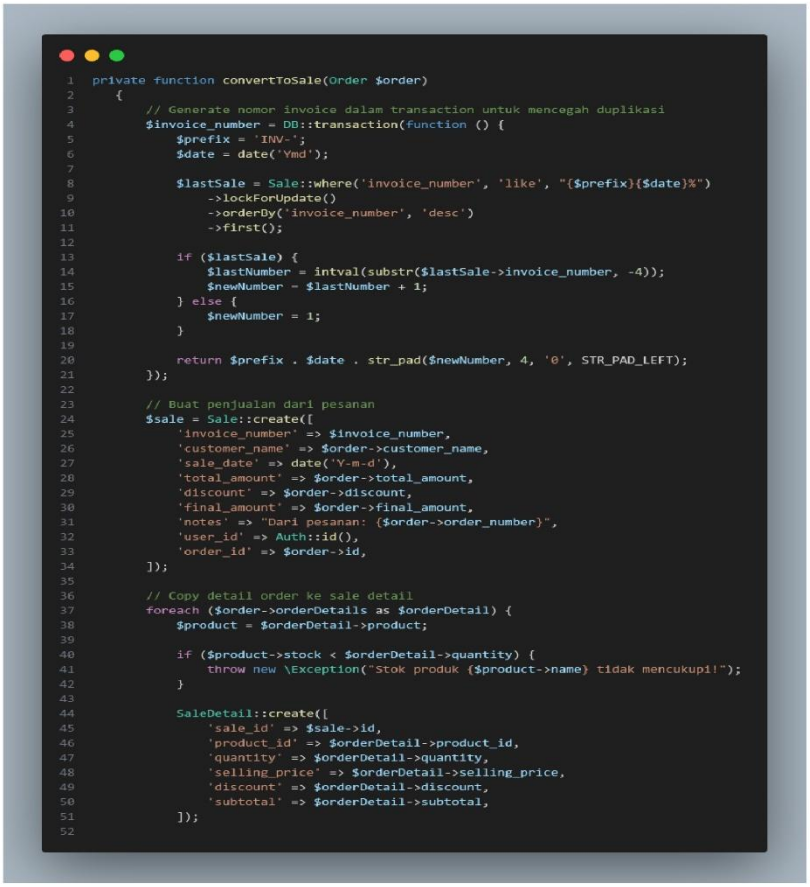


```php
public function update(Request $request, $id)
{
    $order = Order::findOrFail($id);

    $request->validate([
        'order_number' => 'required|string|max:255|unique:orders,order_number,' . $order->id,
        'customer_name' => 'required|string|max:255',
        'customer_phone' => 'nullable|string|min:10|max:13|regex:/^[0-9]{10,13}$/',
        'customer_address' => 'nullable|string',
        'order_date' => 'required|date',
        'down_payment' => 'nullable|numeric|min:0',
        'notes' => 'nullable|string',
        'product_id' => 'required|array',
        'product_id.*' => 'exists:products,id',
        'quantity' => 'required|array',
        'quantity.*' => 'integer|min:1',
        'selling_price' => 'required|array',
        'selling_price.*' => 'numeric|min:0',
        'item_discount' => 'nullable|array',
        'item_discount.*' => 'nullable|numeric|min:0',
        'discount' => 'nullable|numeric|min:0',
    ]);

    // Hitung total dan final amount dengan diskon
    $total_amount = 0;
    foreach ($request->product_id as $key => $value) {
        $item_discount = $request->item_discount[$key] ?? 0;
        $subtotal = ($request->quantity[$key] * $request->selling_price[$key]) - $item_discount;
        $total_amount += $subtotal;
    }

    // Hitung total akhir dengan diskon
    $discount = $request->discount ?? 0;
    $final_amount = $total_amount - $discount;

    DB::beginTransaction();

    try {
        // Update data order
        $order->update([
            'order_number' => $request->order_number,
            'customer_name' => $request->customer_name,
            'customer_phone' => $request->customer_phone,
            'customer_address' => $request->customer_address,
            'order_date' => $request->order_date,
            'total_amount' => $total_amount,
            'discount' => $discount,
            'final_amount' => $final_amount,
            'down_payment' => $request->down_payment ?? 0,
            'notes' => $request->notes,
            'updated_by' => Auth::id(), // Optional: track who updated
        ]);

        // Hapus detail order yang lama
        OrderDetail::where('order_id', $order->id)->delete();

        // Simpan detail order yang baru
        foreach ($request->product_id as $key => $product_id) {
            $quantity = $request->quantity[$key];
            $selling_price = $request->selling_price[$key];
            $item_discount = $request->item_discount[$key] ?? 0;
            $subtotal = ($selling_price * $quantity) - $item_discount;

            OrderDetail::create([
                'order_id' => $order->id,
                'product_id' => $product_id,
                'quantity' => $quantity,
                'selling_price' => $selling_price,
                'discount' => $item_discount,
                'subtotal' => $subtotal,
            ]);
        }
```

- Fungsi ini memperbarui data pesanan. Setelah validasi dan perhitungan ulang total transaksi, sistem akan memperbarui data utama pesanan lalu menghapus semua detail pesanan lama. Detail pesanan yang baru kemudian disimpan ulang.

```php
public function processToSale(Request $request, $id)
{
    $order = Order::findOrFail($id);

    // Check if order can be processed
    if (in_array($order->status, ['completed', 'cancelled'])) {
        return response()->json([
            'success' => false,
            'message' => 'Pesanan dengan status ' . $order->status . ' tidak dapat diproses!'
        ], 422);
    }

    DB::beginTransaction();

    try {
        // Update status order menjadi completed
        $order->update([
            'status' => 'completed',
        ]);

        // Konversi ke penjualan
        $sale = $this->convertToSale($order);

```

- Fungsi ini mengubah status pesanan menjadi selesai (completed) dan mengonversinya menjadi data penjualan (sales). Fungsi ini akan memanggil convertToSale() untuk membuat data baru di tabel sales dan mengurangi stok produk yang terjual.



```php
private function convertToSale(Order $order)
    {
        // Generate nomor invoice dalam transaction untuk mencegah duplikasi
        $invoice_number = DB::transaction(function () {
            $prefix = 'INV-';
            $date = date('Ymd');

            $lastSale = Sale::where('invoice_number', 'like', "{$prefix}{$date}%")
                ->lockForUpdate()
                ->orderBy('invoice_number', 'desc')
                ->first();

            if ($lastSale) {
                $lastNumber = intval(substr($lastSale->invoice_number, -4));
                $newNumber = $lastNumber + 1;
            } else {
                $newNumber = 1;
            }

            return $prefix . $date . str_pad($newNumber, 4, '0', STR_PAD_LEFT);
        });

        // Buat penjualan dari pesanan
        $sale = Sale::create([
            'invoice_number' => $invoice_number,
            'customer_name' => $order->customer_name,
            'sale_date' => date('Y-m-d'),
            'total_amount' => $order->total_amount,
            'discount' => $order->discount,
            'final_amount' => $order->final_amount,
            'notes' => "Dari pesanan: {$order->order_number}",
            'user_id' => Auth::id(),
            'order_id' => $order->id,
        ]);

        // Copy detail order ke sale detail
        foreach ($order->orderDetails as $orderDetail) {
            $product = $orderDetail->product;

            if ($product->stock < $orderDetail->quantity) {
                throw new \Exception("Stok produk {$product->name} tidak mencukupi!");
            }

            SaleDetail::create([
                'sale_id' => $sale->id,
                'product_id' => $orderDetail->product_id,
                'quantity' => $orderDetail->quantity,
                'selling_price' => $orderDetail->selling_price,
                'discount' => $orderDetail->discount,
                'subtotal' => $orderDetail->subtotal,
            ]);

```

- Fungsi ini bertugas membuat penjualan dari data pesanan. Nomor invoice digenerate otomatis. Setiap item dari orderDetails disalin ke sale_details, lalu stok produk dikurangi sesuai jumlah penjualan.

```php
// Method untuk menampilkan halaman print
public function print(Order $order)
{
    $order->load(['user', 'orderDetails.product']);
    return view('pages.pesanan.print', compact('order'));
}

```

- Fungsi print(Order $order)
  Menampilkan halaman cetak HTML dari detail pesanan. Cocok digunakan untuk preview sebelum cetak.

```php
// Method baru untuk download PDF
public function downloadPdf(Order $order)
{
    $order->load(['user', 'orderDetails.product']);

    $pdf = Pdf::loadView('pages.pesanan.pdf', compact('order'));

    // Set paper size dan orientation
    $pdf->setPaper('A4', 'portrait');

    // Set options untuk bahasa Indonesia
    $pdf->setOptions([
        'isHtml5ParserEnabled' => true,
        'isPhpEnabled' => true,
        'defaultFont' => 'DejaVu Sans'
    ]);

    $filename = 'Pesanan_' . $order->order_number . '_' . date('Y-m-d') . '.pdf';

    return $pdf->download($filename);
}
```

- Fungsi ini menghasilkan dan mengunduh file PDF dari data pesanan menggunakan pustaka DomPDF. File diatur agar berukuran A4 dan mendukung font UTF-8 untuk bahasa Indonesia.

```php
public function destroy($id)
{
    DB::beginTransaction();

    try {
        $order = Order::findOrFail($id);

        // Cek apakah order memiliki sale terkait
        $sale = Sale::where('order_id', $order->id)->first();

        if ($sale) {
            // Jika ada sale, hapus data sale tanpa mengembalikan stok
            // Karena stok sudah berkurang saat penjualan terjadi dan tidak perlu dikembalikan

            // Hapus sale details terlebih dahulu
            SaleDetail::where('sale_id', $sale->id)->delete();

            // Hapus sale
            $sale->delete();
        }

        // Hapus order details
        OrderDetail::where('order_id', $order->id)->delete();

        // Hapus order
        $order->delete();

        DB::commit();

        // Check if request is AJAX
        if (request()->ajax()) {
            return response()->json([
                'success' => true,
                'message' => 'Pesanan dan data penjualan terkait berhasil dihapus!'
            ]);
        }

        return redirect()->route('orders.index')
            ->with('success', 'Pesanan dan data penjualan terkait berhasil dihapus!');

    } catch (\Exception $e) {
        DB::rollback();

        // Check if request is AJAX
        if (request()->ajax()) {
            return response()->json([
                'success' => false,
                'message' => 'Terjadi kesalahan saat menghapus data: ' . $e->getMessage()
            ], 500);
        }

        return redirect()->back()
            ->with('error', 'Terjadi kesalahan saat menghapus data: ' . $e->getMessage());
    }
}
```

- Fungsi destroy($id)
  Menghapus data pesanan dan juga penjualannya jika sudah ada. Detail pesanan dan penjualan dihapus terlebih dahulu agar tidak menyisakan data yatim (orphan).

```php
public function sendMessage($sale = null, $order = null)
{
    // Token bisa juga langsung ditulis di sini
    $token = 'vEihVjThDredkzkDrN4v'; // atau: $token = 'YOUR_FONNTE_TOKEN';

    // NOMOR PEMILIK TOKO - Ganti dengan nomor WhatsApp pemilik toko
    $ownerPhone = '6289683659037'; // Nomor pemilik toko yang akan menerima notifikasi

    // Jika tidak ada parameter, ambil data penjualan terbaru dari user yang login
    if (!$sale && !$order) {
        $sale = Sale::with(['saleDetails.product'])
            ->where('user_id', Auth::id())
            ->latest()
            ->first();

        if (!$sale) {
            return response()->json([
                'success' => false,
                'message' => 'Data penjualan tidak ditemukan',
            ]);
        }

        // Ambil data order jika ada
        $order = $sale->order_id ? Order::find($sale->order_id) : null;
    }

    // Build pesan notifikasi untuk pemilik toko
    $message = $this->buildOwnerNotificationMessage($sale, $order);

    // Kirim request ke Fonnte ke nomor pemilik toko
    $response = Http::withHeaders([
        'Authorization' => $token,
    ])->asForm()->post('https://api.fonnte.com/send', [
        'target'  => $ownerPhone,
        'message' => $message,
    ]);

    // Tanggapan
    if ($response->successful()) {
        return response()->json([
            'success' => true,
            'message' => "Notifikasi penjualan berhasil dikirim ke pemilik toko ({$ownerPhone})",
            'response' => $response->json(),
        ]);
    } else {
        return response()->json([
            'success' => false,
            'message' => "Gagal mengirim notifikasi ke pemilik toko ({$ownerPhone})",
            'error' => $response->body(),
        ]);
    }
}

/**
 * Build pesan notifikasi untuk pemilik toko
 */
private function buildOwnerNotificationMessage($sale, $order = null)
{
    $finalAmountFormatted = number_format($sale->final_amount, 0, ',', '.');
    $customerName = $sale->customer_name;
    $saleDate = date('d/m/Y H:i', strtotime($sale->sale_date));
    $staffName = Auth::user()->name ?? 'Staff'; // Nama staff yang memproses

    // Build detail produk
    $productDetails = "";
    $totalItems = 0;

    foreach ($sale->saleDetails as $detail) {
        $subtotalFormatted = number_format($detail->subtotal, 0, ',', '.');
        $productDetails .= "- {$detail->product->name} ({$detail->quantity}x) - Rp {$subtotalFormatted}\n";
        $totalItems += $detail->quantity;
    }

    // Pesan notifikasi untuk pemilik toko
    $message = "🔔 *NOTIFIKASI PENJUALAN BARU*\n\n"
        . "Ada transaksi penjualan baru yang telah diproses!\n\n"
        . "👤 *INFORMASI CUSTOMER:*\n"
        . "📛 Nama: {$customerName}\n";

    // Tambahkan nomor customer jika ada
    if ($order && $order->customer_phone) {
        $message .= "📱 Telepon: {$order->customer_phone}\n";
    }

    $message .= "\n📋 *DETAIL TRANSAKSI:*\n"
        . "📝 No. Invoice: {$sale->invoice_number}\n"
        . "🗓️ Tanggal: {$saleDate}\n"
        . "👨‍💼 Diproses oleh: {$staffName}\n"
        . "📦 Total Item: {$totalItems} produk\n\n"
        . "🛍️ *PRODUK TERJUAL:*\n"
        . $productDetails . "\n"
        . "💰 *TOTAL PENJUALAN: Rp {$finalAmountFormatted}*\n\n";

    // Tambahkan info DP dan sisa bayar jika ada data order
    if ($order && $order->down_payment > 0) {
        $downPaymentFormatted = number_format($order->down_payment, 0, ',', '.');
        $remainingAmount = $sale->final_amount - $order->down_payment;
        $remainingFormatted = number_format($remainingAmount, 0, ',', '.');

        $message .= "💳 DP Diterima: Rp {$downPaymentFormatted}\n"
            . "💸 Sisa tagihan: Rp {$remainingFormatted}\n\n";
    }

    $message .= "✅ Status: TRANSAKSI BERHASIL\n\n"
        . "📊 Silakan cek laporan penjualan untuk detail lengkap.\n\n"
        . "🎉 Selamat! Penjualan hari ini semakin meningkat! 💪";

    return $message;
}
```

- Fungsi sendMessage()
  Fungsi ini mengirim pesan WhatsApp ke pemilik toko melalui API Fonnte, berisi notifikasi tentang penjualan yang berhasil. Data pengiriman diambil dari penjualan terbaru atau bisa juga dari parameter yang dikirim.

```php
public function cancel(Request $request, $id)
    {
        $request->validate([
            'reason' => 'nullable|string|max:500',
        ]);

        $order = Order::findOrFail($id);

        // Check if order can be cancelled
        if (in_array($order->status, ['completed', 'cancelled'])) {
            return response()->json([
                'success' => false,
                'message' => 'Pesanan dengan status ' . $order->status . ' tidak dapat dibatalkan!'
            ], 422);
        }

        DB::beginTransaction();

        try {
            // Update order status to cancelled
            $order->update([
                'status' => 'cancelled',
                'notes' => $order->notes . "\n\nDibatalkan pada " . now()->format('d/m/Y H:i') .
                           ($request->reason ? ". Alasan: " . $request->reason : "")
            ]);

            DB::commit();

            return response()->json([
                'success' => true,
                'message' => 'Pesanan berhasil dibatalkan!'
            ]);
        } catch (\Exception $e) {
            DB::rollback();

            return response()->json([
                'success' => false,
                'message' => 'Terjadi kesalahan: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi ini digunakan untuk membatalkan pesanan. Jika pesanan belum dalam status completed atau cancelled, maka statusnya akan diubah menjadi cancelled dan catatan pembatalan disimpan.

MODEL SALE

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Sale extends Model
{
    use HasFactory;

    protected $fillable = [
        'invoice_number',
        'customer_name',
        'sale_date',
        'total_amount',
        'discount',
        'final_amount',
        'notes',
        'user_id',
        'order_id', // Tambahkan kolom ini
    ];

    protected $casts = [
        'sale_date' => 'date',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function saleDetails()
    {
        return $this->hasMany(SaleDetail::class);
    }

    public function order()
    {
        return $this->belongsTo(Order::class);
    }
}
```

- protected $fillable
  Daftar kolom ini merupakan field yang diizinkan untuk di-*mass assign*, artinya kolom-kolom tersebut bisa langsung diisi dari input pengguna menggunakan method seperti create() atau update(). order_id menandakan bahwa setiap entri penjualan dapat dikaitkan ke satu pesanan.
- public function user()
  Relasi ini menunjukkan bahwa penjualan dilakukan oleh seorang pengguna. Foreign key user_id menghubungkan model Sale dengan model User.
- public function saleDetails()
  Setiap penjualan dapat memiliki beberapa detail penjualan. Relasi ini menghubungkan model Sale dengan banyak entri di SaleDetail, yang berisi informasi seperti produk, kuantitas, harga jual, dan diskon.
- public function order()
  Fungsi ini menyatakan bahwa setiap penjualan mungkin berasal dari satu pesanan (Order). Relasi belongsTo digunakan karena kolom order_id ada di tabel sales. Dengan ini, kita bisa mengambil data pesanan sumber dari sebuah penjualan.

SALE CONTROLER

```php
class SaleController extends Controller
{
    /**
     * Menampilkan daftar penjualan
     * Semua penjualan berasal dari pesanan yang telah selesai
     */
    public function index()
    {
        $sales = Sale::with(['user', 'order'])
            ->orderBy('sale_date', 'desc')
            ->latest()
            ->get();

        return view('pages.penjualan.index', compact('sales'));
    }
```

- Fungsi ini menampilkan daftar semua data penjualan yang tersimpan di database. Pengambilan data dilakukan menggunakan Eloquent dan memuat relasi user (yang memproses penjualan) serta order (pesanan asal). Penjualan diurutkan berdasarkan tanggal penjualan terbaru dengan metode orderBy() dan latest(), kemudian dikirim ke view pages.penjualan.index sebagai variabel sales.

```php
public function show(Sale $sale)
    {
        $sale->load(['user', 'saleDetails.product', 'order']);

        return view('pages.penjualan.detail', compact('sale'));
    }
    //  /**
    //  * Mencetak faktur penjualan (invoice)
    //  */
    public function invoice(Sale $sale)
    {
        $sale->load(['user', 'saleDetails.product', 'order']);

        return view('pages.penjualan.invoice', compact('sale'));
    }

```

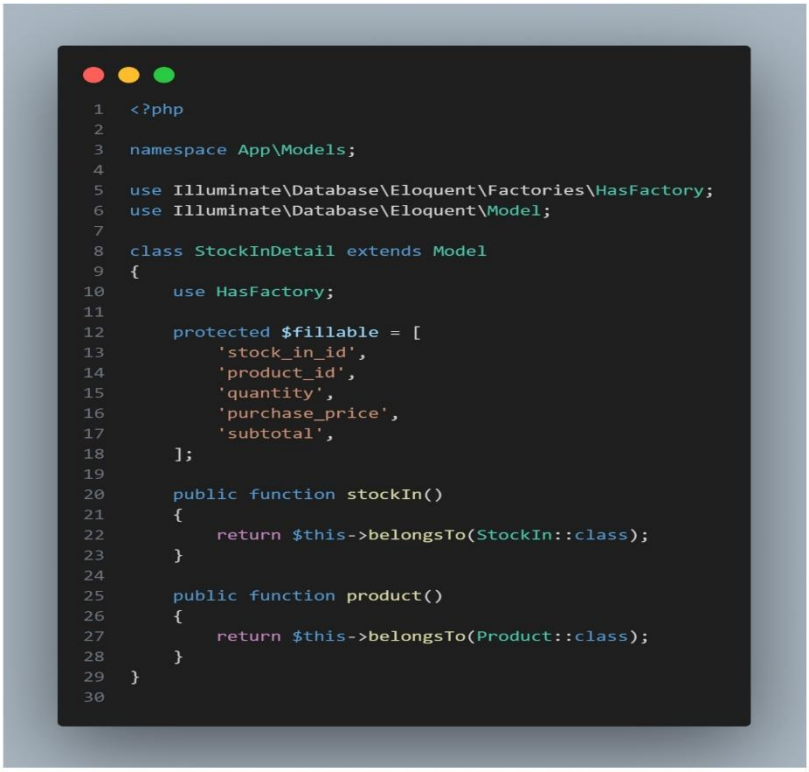
- public function show(Sale $sale)
  Fungsi show() digunakan untuk menampilkan detail lengkap dari satu transaksi penjualan tertentu. Data yang diambil mencakup relasi ke user, daftar saleDetails (produk yang dijual beserta kuantitas, harga, diskon, dll), serta informasi asal penjualan yaitu order. Data ini kemudian dikirim ke tampilan detail untuk ditampilkan kepada pengguna.
- public function invoice(Sale $sale)
  Fungsi ini menghasilkan tampilan faktur penjualan (invoice) untuk penjualan tertentu. Sama seperti show(), data dimuat lengkap dengan semua relasi yang dibutuhkan. View invoice biasanya dirancang dalam format siap cetak atau PDF yang mencantumkan informasi transaksi, pelanggan, produk, dan total pembayaran.

MODEL STOK

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class StockIn extends Model
{
    use HasFactory;

    protected $fillable = [
        'reference_number',
        'date',
        'source',
        'total_amount',
        'notes',
        'user_id',
    ];

    protected $casts = [
        'date' => 'date',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function stockInDetails()
    {
        return $this->hasMany(StockInDetail::class);
    }
}

```

- protected $fillable
  Properti ini menentukan field apa saja yang boleh diisi secara massal menggunakan method seperti create() atau update(). Field reference_number digunakan sebagai kode referensi stok masuk, source dapat berisi nama supplier atau sumber barang, dan total_amount adalah nilai total stok yang masuk. Kolom user_id menyimpan ID pengguna yang melakukan pencatatan.
- protected $casts
  Properti ini memberi tahu Laravel bahwa kolom date harus dikonversi menjadi objek Carbon. Dengan demikian, tanggal dapat dimanipulasi dan diformat dengan mudah menggunakan fungsi waktu Laravel seperti format(), diffInDays(), dan lain-lain.
- public function user()
  Relasi ini menunjukkan bahwa pencatatan stok masuk dilakukan oleh satu pengguna. Foreign key user_id digunakan untuk menghubungkan stok masuk ke model User.
- public function stockInDetails()
  Fungsi ini menyatakan bahwa satu entri StockIn dapat memiliki banyak baris detail (StockInDetail). Setiap detail akan menyimpan informasi produk, kuantitas, harga per unit, dan subtotal. Relasi ini penting untuk pelacakan stok barang secara akurat per item yang masuk.

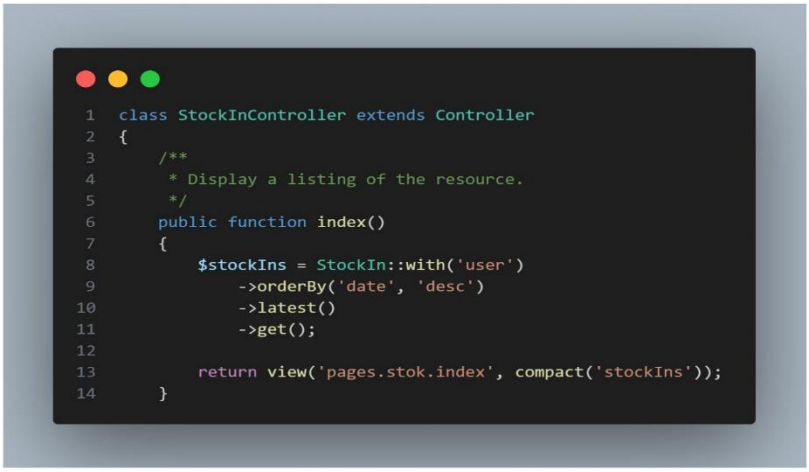
MODEL STOK DETAIL



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class StockInDetail extends Model
{
    use HasFactory;

    protected $fillable = [
        'stock_in_id',
        'product_id',
        'quantity',
        'purchase_price',
        'subtotal',
    ];

    public function stockIn()
    {
        return $this->belongsTo(StockIn::class);
    }

    public function product()
    {
        return $this->belongsTo(Product::class);
    }
}

```

- protected $fillable
  Properti ini mendefinisikan kolom-kolom yang dapat di-*mass assign*. Artinya, hanya kolom-kolom ini yang bisa langsung diisi dari form input atau saat membuat data dengan create(). Field quantity menunjukkan jumlah barang yang masuk, purchase_price menyimpan harga beli per unit, dan subtotal adalah hasil perkalian kuantitas dengan harga.
- public function stockIn()
  Fungsi ini mendefinisikan bahwa setiap baris detail stok masuk (StockInDetail) dimiliki oleh satu entri utama stok masuk (StockIn). Ini adalah relasi belongsTo, dengan foreign key stock_in_id sebagai penghubung.
- public function product()
  Relasi ini menghubungkan baris detail ke entitas produk yang dimaksud. Setiap detail akan menunjukkan produk mana yang masuk, sehingga kamu dapat mengakses data produk seperti nama, satuan, atau stok saat ini dengan mudah melalui $stockInDetail->product.

STOK CONTROLLER

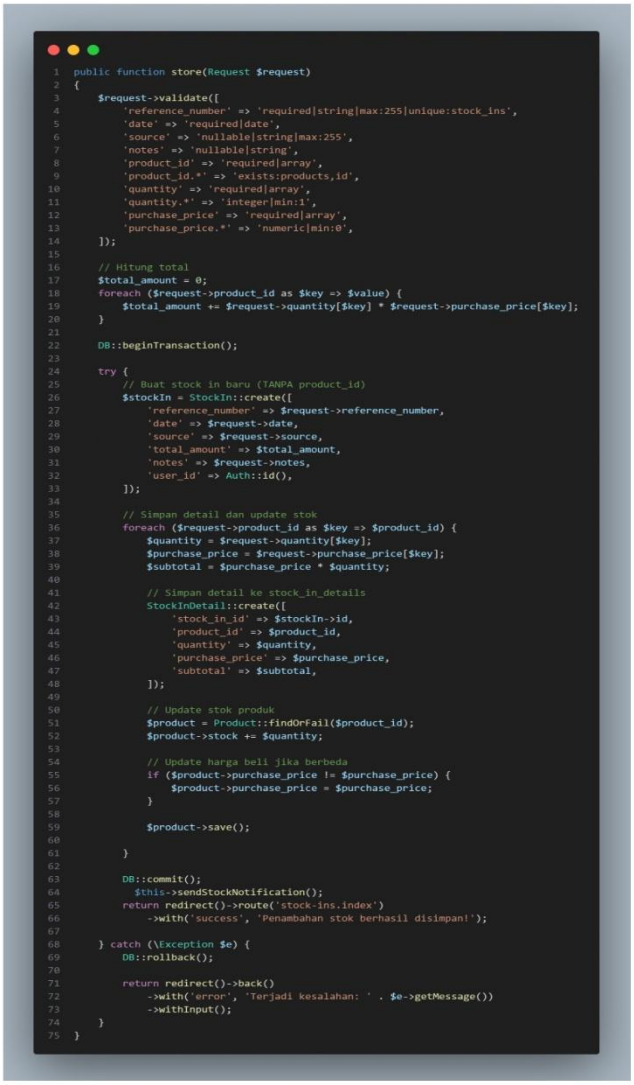


```php
class StockInController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $stockIns = StockIn::with('user')
            ->orderBy('date', 'desc')
            ->latest()
            ->get();

        return view('pages.stok.index', compact('stockIns'));
    }
```

- Fungsi ini mengambil semua data stok masuk (StockIn) dari database, memuat relasi user, lalu mengurutkannya berdasarkan tanggal terbaru. Data ditampilkan melalui tampilan pages.stok.index.

```php
public function create()
    {
        $products = Product::all();

        // Generate nomor referensi otomatis
        $prefix = 'STK-IN-';
        $date = date('Ymd');
        $lastStockIn = StockIn::where('reference_number', 'like', "{$prefix}{$date}%")
            ->orderBy('reference_number', 'desc')
            ->first();

        if ($lastStockIn) {
            $lastNumber = intval(substr($lastStockIn->reference_number, -4));
            $newNumber = $lastNumber + 1;
        } else {
            $newNumber = 1;
        }

        $reference_number = $prefix . $date . str_pad($newNumber, 4, '0', STR_PAD_LEFT);

        return view('pages.stok.tambah-stok', compact('products', 'reference_number'));
    }
    /**
```

- Fungsi ini menampilkan form tambah stok dan membuat nomor referensi otomatis dengan format STK-IN-{tanggal}{nomor urut}. Sistem mengecek entri terakhir hari itu untuk menentukan nomor berikutnya, sehingga menghindari duplikasi.



```php
public function store(Request $request)
{
    $request->validate([
        'reference_number' => 'required|string|max:255|unique:stock_ins',
        'date' => 'required|date',
        'source' => 'nullable|string|max:255',
        'notes' => 'nullable|string',
        'product_id' => 'required|array',
        'product_id.*' => 'exists:products,id',
        'quantity' => 'required|array',
        'quantity.*' => 'integer|min:1',
        'purchase_price' => 'required|array',
        'purchase_price.*' => 'numeric|min:0',
    ]);

    // Hitung total
    $total_amount = 0;
    foreach ($request->product_id as $key => $value) {
        $total_amount += $request->quantity[$key] * $request->purchase_price[$key];
    }

    DB::beginTransaction();

    try {
        // Buat stock in baru (TANPA product_id)
        $stockIn = StockIn::create([
            'reference_number' => $request->reference_number,
            'date' => $request->date,
            'source' => $request->source,
            'total_amount' => $total_amount,
            'notes' => $request->notes,
            'user_id' => Auth::id(),
        ]);

        // Simpan detail dan update stok
        foreach ($request->product_id as $key => $product_id) {
            $quantity = $request->quantity[$key];
            $purchase_price = $request->purchase_price[$key];
            $subtotal = $purchase_price * $quantity;

            // Simpan detail ke stock_in_details
            StockInDetail::create([
                'stock_in_id' => $stockIn->id,
                'product_id' => $product_id,
                'quantity' => $quantity,
                'purchase_price' => $purchase_price,
                'subtotal' => $subtotal,
            ]);

            // Update stok produk
            $product = Product::findOrFail($product_id);
            $product->stock += $quantity;

            // Update harga beli jika berbeda
            if ($product->purchase_price != $purchase_price) {
                $product->purchase_price = $purchase_price;
            }

            $product->save();

        }

        DB::commit();
          $this->sendStockNotification();
        return redirect()->route('stock-ins.index')
            ->with('success', 'Penambahan stok berhasil disimpan!');

    } catch (\Exception $e) {
        DB::rollback();

        return redirect()->back()
            ->with('error', 'Terjadi kesalahan: ' . $e->getMessage())
            ->withInput();
    }
}
```

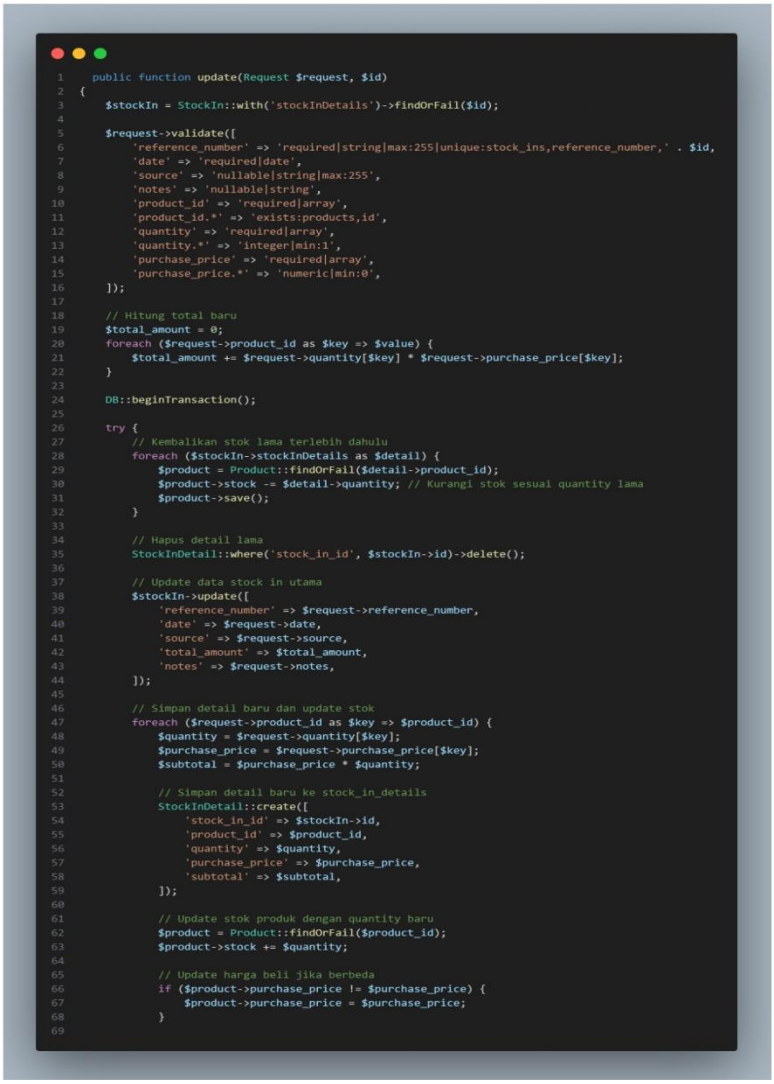
- Fungsi ini menyimpan data stok masuk. Setelah validasi form, sistem menghitung total nilai barang masuk, lalu menyimpan data utama ke tabel `stock_ins`. Untuk setiap produk, sistem juga menyimpan detailnya ke `stock_in_details`, menambahkan jumlah stok, dan memperbarui harga beli jika berubah. Semua proses dilakukan dalam transaksi database agar konsisten dan aman.

```php
public function show(StockIn $stockIn)
    {
        $stockIn->load(['user', 'stockInDetails.product']);

        return view('pages.stok.detail-stok', compact('stockIn'));
    }
```

- Fungsi show()
  Menampilkan detail satu entri stok masuk lengkap dengan informasi produk, jumlah, harga beli, dan nama pengguna yang memproses.

```php
public function edit($id)
{
    $stockIn = StockIn::with(['stockInDetails.product', 'user'])->findOrFail($id);
    $products = Product::all();
     return view('pages.stok.edit-stock', compact('stockIn', 'products'));
    }
```

- Fungsi ini digunakan untuk menampilkan form edit stok masuk. Data lengkap, termasuk detail produk dan pengguna, dikirim ke view edit-stock.

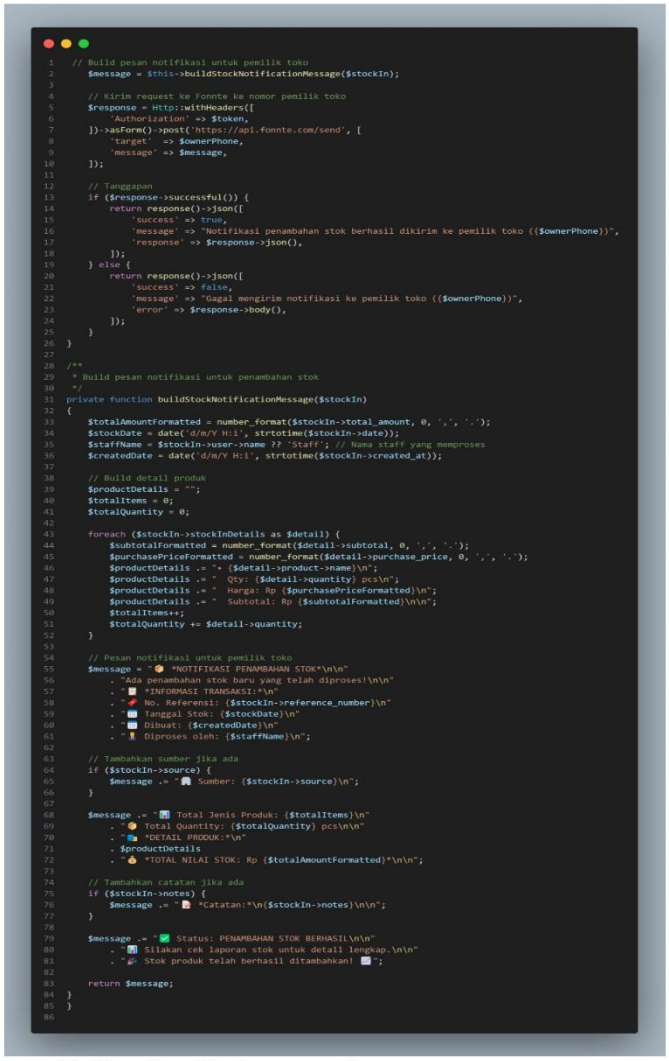


```php
public function update(Request $request, $id)
{
    $stockIn = StockIn::with('stockInDetails')->findOrFail($id);

    $request->validate([
        'reference_number' => 'required|string|max:255|unique:stock_ins,reference_number,' . $id,
        'date' => 'required|date',
        'source' => 'nullable|string|max:255',
        'notes' => 'nullable|string',
        'product_id' => 'required|array',
        'product_id.*' => 'exists:products,id',
        'quantity' => 'required|array',
        'quantity.*' => 'integer|min:1',
        'purchase_price' => 'required|array',
        'purchase_price.*' => 'numeric|min:0',
    ]);

    // Hitung total baru
    $total_amount = 0;
    foreach ($request->product_id as $key => $value) {
        $total_amount += $request->quantity[$key] * $request->purchase_price[$key];
    }

    DB::beginTransaction();

    try {
        // Kembalikan stok lama terlebih dahulu
        foreach ($stockIn->stockInDetails as $detail) {
            $product = Product::findOrFail($detail->product_id);
            $product->stock -= $detail->quantity; // Kurangi stok sesuai quantity lama
            $product->save();
        }

        // Hapus detail lama
        StockInDetail::where('stock_in_id', $stockIn->id)->delete();

        // Update data stock in utama
        $stockIn->update([
            'reference_number' => $request->reference_number,
            'date' => $request->date,
            'source' => $request->source,
            'total_amount' => $total_amount,
            'notes' => $request->notes,
        ]);

        // Simpan detail baru dan update stok
        foreach ($request->product_id as $key => $product_id) {
            $quantity = $request->quantity[$key];
            $purchase_price = $request->purchase_price[$key];
            $subtotal = $purchase_price * $quantity;

            // Simpan detail baru ke stock_in_details
            StockInDetail::create([
                'stock_in_id' => $stockIn->id,
                'product_id' => $product_id,
                'quantity' => $quantity,
                'purchase_price' => $purchase_price,
                'subtotal' => $subtotal,
            ]);

            // Update stok produk dengan quantity baru
            $product = Product::findOrFail($product_id);
            $product->stock += $quantity;

            // Update harga beli jika berbeda
            if ($product->purchase_price != $purchase_price) {
                $product->purchase_price = $purchase_price;
            }

```

- Fungsi update()
  Saat melakukan pembaruan data stok masuk, sistem pertama-tama mengembalikan stok produk sesuai data lama, lalu menghapus semua detail lama. Setelah itu, data baru disimpan dan stok diperbarui kembali dengan nilai yang baru. Ini memastikan stok tetap akurat meskipun ada perubahan data.

```php
public function sendStockNotification($stockIn = null)
{
    // Token Fonnte
    $token = 'vEHvVjThDredkzkDrN4v'; // Ganti dengan token Fonnte Anda

    // NOMOR PEMILIK TOKO - Ganti dengan nomor WhatsApp pemilik toko
    $ownerPhone = '6289603636937'; // Nomor pemilik toko yang akan menerima notifikasi

    // Jika tidak ada parameter, ambil data penambahan stok terbaru dari user yang login
    if (!$stockIn) {
        $stockIn = StockIn::with(['stockInDetails.product', 'user'])
            ->where('user_id', Auth::id())
            ->latest()
            ->first();

        if (!$stockIn) {
            return response()->json([
                'success' => false,
                'message' => 'Data penambahan stok tidak ditemukan',
            ]);
        }
    }
```

- Fungsi sendStockNotification()
  Fungsi ini mengirimkan notifikasi WhatsApp ke pemilik toko menggunakan Fonnte setelah stok berhasil ditambahkan. Sistem mengambil data stok terbaru yang dibuat oleh user, lalu membentuk pesan yang informatif dan rapi.



```php
// Build pesan notifikasi untuk pemilik toko
    $message = $this->buildStockNotificationMessage($stockIn);

    // Kirim request ke Fonnte ke nomor pemilik toko
    $response = Http::withHeaders([
        'Authorization' => $token,
    ])->asForm()->post('https://api.fonnte.com/send', [
        'target'  => $ownerPhone,
        'message' => $message,
    ]);

    // Tanggapan
    if ($response->successful()) {
        return response()->json([
            'success' => true,
            'message' => "Notifikasi penambahan stok berhasil dikirim ke pemilik toko ({$ownerPhone})",
            'response' => $response->json(),
        ]);
    } else {
        return response()->json([
            'success' => false,
            'message' => "Gagal mengirim notifikasi ke pemilik toko ({$ownerPhone})",
            'error' => $response->body(),
        ]);
    }
}

/**
 * Build pesan notifikasi untuk penambahan stok
 */
private function buildStockNotificationMessage($stockIn)
{
    $totalAmountFormatted = number_format($stockIn->total_amount, 0, ',', '.');
    $stockDate = date('d/m/Y H:i', strtotime($stockIn->date));
    $staffName = $stockIn->user->name ?? 'Staff'; // Nama staff yang memproses
    $createdDate = date('d/m/Y H:i', strtotime($stockIn->created_at));

    // Build detail produk
    $productDetails = "";
    $totalItems = 0;
    $totalQuantity = 0;

    foreach ($stockIn->stockInDetails as $detail) {
        $subtotalFormatted = number_format($detail->subtotal, 0, ',', '.');
        $purchasePriceFormatted = number_format($detail->purchase_price, 0, ',', '.');
        $productDetails .= "• {$detail->product->name}\n";
        $productDetails .= "  Qty: {$detail->quantity} pcs\n";
        $productDetails .= "  Harga: Rp {$purchasePriceFormatted}\n";
        $productDetails .= "  Subtotal: Rp {$subtotalFormatted}\n\n";
        $totalItems++;
        $totalQuantity += $detail->quantity;
    }

    // Pesan notifikasi untuk pemilik toko
    $message = "📦 *NOTIFIKASI PENAMBAHAN STOK*\n\n"
        . "Ada penambahan stok baru yang telah diproses!\n\n"
        . "📋 *INFORMASI TRANSAKSI:*\n"
        . "🔖 No. Referensi: {$stockIn->reference_number}\n"
        . "📅 Tanggal Stok: {$stockDate}\n"
        . "🗓️ Dibuat: {$createdDate}\n"
        . "👤 Diproses oleh: {$staffName}\n";

    // Tambahkan sumber jika ada
    if ($stockIn->source) {
        $message .= "🏭 Sumber: {$stockIn->source}\n";
    }

    $message .= "\n📊 Total Jenis Produk: {$totalItems}\n"
        . "📦 Total Quantity: {$totalQuantity} pcs\n\n"
        . "📝 *DETAIL PRODUK:*\n"
        . $productDetails
        . "💰 *TOTAL NILAI STOK: Rp {$totalAmountFormatted}*\n\n";

    // Tambahkan catatan jika ada
    if ($stockIn->notes) {
        $message .= "📄 *Catatan:*\n{$stockIn->notes}\n\n";
    }

    $message .= "✅ Status: PENAMBAHAN STOK BERHASIL\n\n"
        . "📊 Silakan cek laporan stok untuk detail lengkap.\n\n"
        . "🎉 Stok produk telah berhasil ditambahkan! 📈";

    return $message;
}
}
```

- Fungsi buildStockNotificationMessage()
  Fungsi ini menyusun isi pesan notifikasi WhatsApp yang akan dikirim. Informasi dalam pesan mencakup: nomor referensi, tanggal, sumber barang, nama staff, detail produk (nama, qty, harga, subtotal), total nilai stok, dan catatan tambahan jika ada. Format ini membuat pesan mudah dibaca dan profesional.

FINANCIAL REPORT CONTROLLER

```php
class FinancialReportController extends Controller
{
    /**
     * 1. Laporan Laba Rugi (Income Statement)
     */
    public function profitLossReport(Request $request)
    {
        $startDate = $request->input('start_date', Carbon::now()->startOfMonth()->format('Y-m-d'));
        $endDate = $request->input('end_date', Carbon::now()->endOfMonth()->format('Y-m-d'));

        // Total Pendapatan
        $totalRevenue = DB::table('sales')
            ->whereBetween('sale_date', [$startDate, $endDate])
            ->sum('final_amount');

        // Total HPP
        $totalCogs = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales as s', 'sd.sale_id', '=', 's.id')
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->sum(DB::raw('sd.quantity * p.purchase_price'));

        // Gross Profit
        $grossProfit = $totalRevenue - $totalCogs;

        // Detail per bulan
      $monthlyData = DB::table('sales as s')
    ->leftJoin('sale_details as sd', 's.id', '=', 'sd.sale_id')
    ->leftJoin('products as p', 'sd.product_id', '=', 'p.id')
    ->select(
        DB::raw('YEAR(s.sale_date) as year'),
        DB::raw('MONTH(s.sale_date) as month'),
        DB::raw('SUM(DISTINCT s.final_amount) as revenue'), // DISTINCT untuk menghindari duplikasi
        DB::raw('SUM(sd.quantity * p.purchase_price) as cogs'),
        DB::raw('SUM(DISTINCT s.final_amount) - SUM(sd.quantity * p.purchase_price) as gross_profit')
    )
    ->whereBetween('s.sale_date', [$startDate, $endDate])
    ->groupBy(DB::raw('YEAR(s.sale_date)'), DB::raw('MONTH(s.sale_date)'))
    ->orderBy('year')
    ->orderBy('month')
    ->get();

        return view('pages.FinancialReport.Laporan-Laba-Rugi', compact(
            'startDate', 'endDate', 'totalRevenue', 'totalCogs',
            'grossProfit', 'monthlyData'
        ));
    }
```

- Fungsi ini menyajikan laporan laba rugi berdasarkan periode waktu yang dipilih (default: bulan berjalan). Total pendapatan dihitung dari sales.final_amount, sementara HPP (harga pokok penjualan) dihitung dari sale_details.quantity * products.purchase_price. Gross profit diperoleh dengan mengurangkan HPP dari pendapatan. Data juga dikelompokkan per bulan menggunakan fungsi agregasi SQL dan ditampilkan dalam view Laporan-Laba-Rugi.

```php
public function cogsReport(Request $request)
    {
        $startDate = $request->input('start_date', Carbon::now()->startOfMonth()->format('Y-m-d'));
        $endDate = $request->input('end_date', Carbon::now()->endOfMonth()->format('Y-m-d'));

        $cogsDetail = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales as s', 'sd.sale_id', '=', 's.id')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                'p.name as product_name',
                'p.purchase_price',
                DB::raw('SUM(sd.quantity) as total_sold'),
                DB::raw('SUM(sd.quantity * p.purchase_price) as total_cogs')
            )
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->groupBy('c.name', 'p.id', 'p.name', 'p.purchase_price')
            ->orderBy('total_cogs', 'desc')
            ->get();

        $cogsByCategory = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales as s', 'sd.sale_id', '=', 's.id')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                DB::raw('SUM(sd.quantity * p.purchase_price) as category_cogs')
            )
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->groupBy('c.id', 'c.name')
            ->orderBy('category_cogs', 'desc')
            ->get();

        $totalCogs = $cogsDetail->sum('total_cogs');

        return view('pages.FinancialReport.Laporan-Harga-Pokok-Penjualan', compact(
            'startDate', 'endDate', 'totalCogs', 'cogsByCategory', 'cogsDetail'
        ));
    }
```

- Fungsi Laporan ini memberikan rincian harga pokok penjualan per produk dan per kategori. Dengan bergabung ke tabel categories, hasilnya menampilkan produk, harga beli, total penjualan, dan total nilai HPP. Laporan ini juga menghitung total keseluruhan HPP dan urutan kategori berdasarkan HPP tertinggi. Output dikirim ke view Laporan-Harga-Pokok-Penjualan.

```php
public function grossProfitReport(Request $request)
    {
        $startDate = $request->input('start_date', Carbon::now()->startOfMonth()->format('Y-m-d'));
        $endDate = $request->input('end_date', Carbon::now()->endOfMonth()->format('Y-m-d'));

        // Gross profit per produk
        $productProfitability = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales as s', 'sd.sale_id', '=', 's.id')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                'p.name as product_name',
                'p.purchase_price',
                'p.selling_price',
                DB::raw('SUM(sd.quantity) as total_sold'),
                DB::raw('SUM(sd.subtotal) as total_revenue'),
                DB::raw('SUM(sd.quantity * p.purchase_price) as total_cogs'),
                DB::raw('SUM(sd.subtotal) - SUM(sd.quantity * p.purchase_price) as gross_profit'),
                DB::raw('ROUND(((p.selling_price - p.purchase_price) / p.purchase_price) * 100, 2) as margin_percent')
            )
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->groupBy('c.name', 'p.id', 'p.name', 'p.purchase_price', 'p.selling_price')
            ->orderBy('gross_profit', 'desc')
            ->get();

        // Gross profit harian
        $dailyGrossProfit = DB::table('sales as s')
            ->join('sale_details as sd', 's.id', '=', 'sd.sale_id')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->select(
                's.sale_date',
                DB::raw('SUM(s.final_amount) as daily_revenue'),
                DB::raw('SUM(sd.quantity * p.purchase_price) as daily_cogs'),
                DB::raw('SUM(s.final_amount) - SUM(sd.quantity * p.purchase_price) as daily_gross_profit')
            )
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->groupBy('s.sale_date')
            ->orderBy('s.sale_date')
            ->get();

        $totalGrossProfit = $productProfitability->sum('gross_profit');
        $averageMargin = $productProfitability->avg('margin_percent');

        return view('pages.FinancialReport.Laporan-Gross-Profit', compact(
            'startDate', 'endDate', 'productProfitability', 'dailyGrossProfit',
            'totalGrossProfit', 'averageMargin'
        ));
```

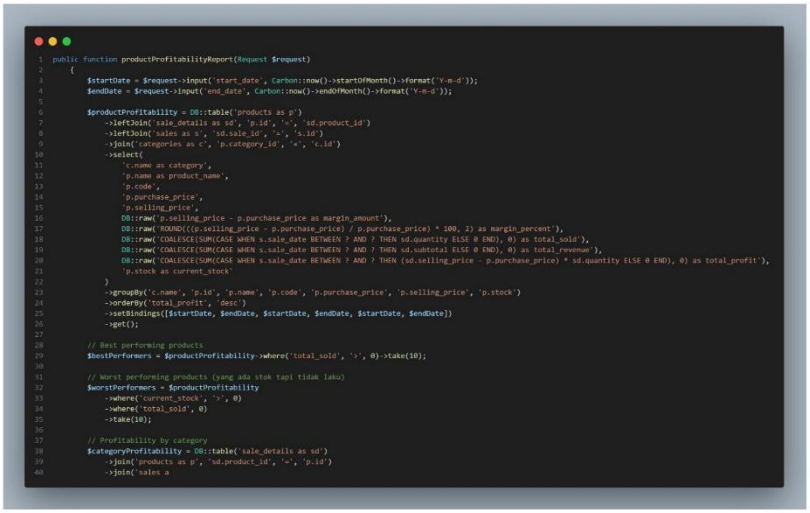
- Fungsi ini menghitung profit kotor per produk dan per hari. Data mencakup margin keuntungan, harga jual vs harga beli, jumlah unit terjual, dan total revenue. Selain itu, laporan ini juga menyajikan grafik harian laba kotor, serta margin rata-rata sebagai ukuran efisiensi. Hasilnya ditampilkan dalam view Laporan-Gross-Profit.

```php
public function salesDetailReport(Request $request)
    {
    $startDate = $request->input('start_date', Carbon::now()->startOfMonth()->format('Y-m-d'));
    $endDate = $request->input('end_date', Carbon::now()->endOfMonth()->format('Y-m-d'));

    // Penjualan per produk
    $salesByProduct = DB::table('sale_details as sd')
        ->join('products as p', 'sd.product_id', '=', 'p.id')
        ->join('categories as c', 'p.category_id', '=', 'c.id')
        ->join('sales as s', 'sd.sale_id', '=', 's.id')
        ->select(
            'c.name as category',
            'p.name as product_name',
            'p.unit',
            DB::raw('SUM(sd.quantity) as total_sold'),
            DB::raw('SUM(sd.subtotal) as total_revenue'),
            DB::raw('AVG(sd.selling_price) as avg_selling_price'),
            DB::raw('COUNT(DISTINCT s.id) as transaction_count')
        )
        ->whereBetween('s.sale_date', [$startDate, $endDate])
        ->groupBy('c.name', 'p.id', 'p.name', 'p.unit')
        ->orderBy('total_revenue', 'desc')
        ->get();

    // Penjualan per kategori
    $salesByCategory = DB::table('sale_details as sd')
        ->join('products as p', 'sd.product_id', '=', 'p.id')
        ->join('categories as c', 'p.category_id', '=', 'c.id')
        ->join('sales as s', 'sd.sale_id', '=', 's.id')
        ->select(
            'c.name as category',
            DB::raw('SUM(sd.quantity) as total_quantity'),
            DB::raw('SUM(sd.subtotal) as total_revenue'),
            DB::raw('COUNT(DISTINCT p.id) as product_variety')
        )
        ->whereBetween('s.sale_date', [$startDate, $endDate])
        ->groupBy('c.id', 'c.name')
        ->orderBy('total_revenue', 'desc')
        ->get();

    // Top customer
    $topCustomers = DB::table('sales')
        ->select(
            'customer_name',
            DB::raw('COUNT(*) as transaction_count'),
            DB::raw('SUM(final_amount) as total_purchase')
        )
        ->whereBetween('sale_date', [$startDate, $endDate])
        ->whereNotNull('customer_name')
        ->where('customer_name', '!=', '')
        ->groupBy('customer_name')
        ->orderBy('total_purchase', 'desc')
        ->limit(10)
        ->get();

    // PERBAIKAN: Hitung total revenue langsung dari tabel sales (konsisten)
    $totalRevenue = DB::table('sales')
        ->whereBetween('sale_date', [$startDate, $endDate])
        ->sum('final_amount');

    $totalTransactions = DB::table('sales')
        ->whereBetween('sale_date', [$startDate, $endDate])
        ->count();
```

- Menampilkan statistik penjualan per produk, per kategori, dan pelanggan terbaik. Dihitung juga total transaksi, total pendapatan, dan rata-rata transaksi. Laporan ini membantu mengidentifikasi produk dengan revenue tertinggi dan pelanggan paling aktif. View terkait: Laporan-Sale-Detail.

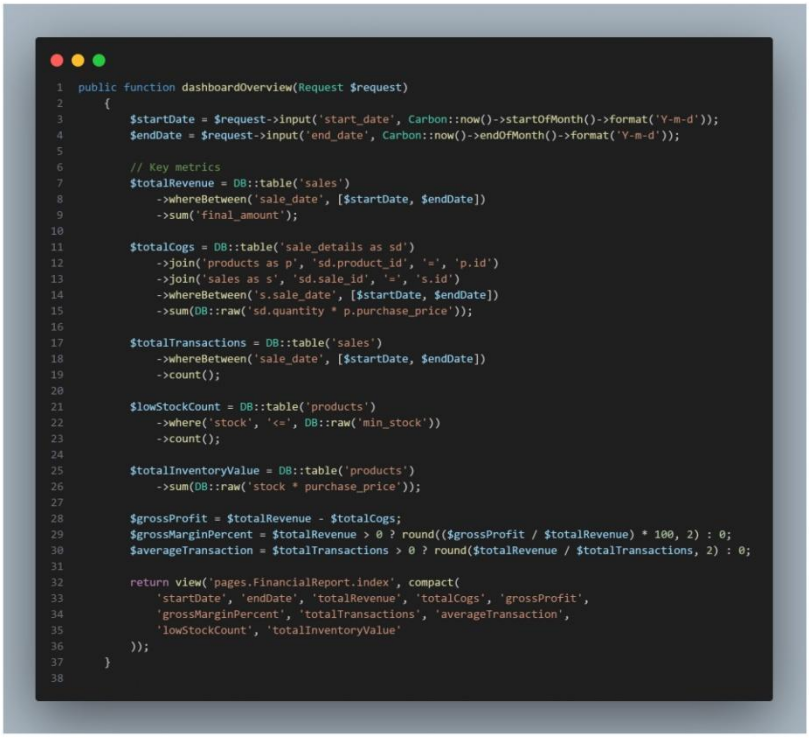
```php
public function inventoryReport(Request $request)
    {
        // Current inventory value
        $currentInventory = DB::table('products as p')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                'p.name as product_name',
                'p.code',
                'p.stock as current_stock',
                'p.min_stock',
                'p.unit',
                'p.purchase_price',
                DB::raw('p.stock * p.purchase_price as inventory_value'),
                DB::raw('CASE WHEN p.stock <= p.min_stock THEN "Low Stock" ELSE "Normal" END as stock_status')
            )
            ->orderBy('c.name')
            ->orderBy('p.name')
            ->get();

        // Low stock products
        $lowStockProducts = DB::table('products as p')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                'p.name as product_name',
                'p.stock as current_stock',
                'p.min_stock',
                DB::raw('p.min_stock - p.stock as need_to_order'),
                DB::raw('(p.min_stock - p.stock + 10) * p.purchase_price as estimated_cost')
            )
            ->where('p.stock', '<=', DB::raw('p.min_stock'))
            ->orderBy('p.stock')
            ->get();

        // Inventory by category
        $inventoryByCategory = DB::table('products as p')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                DB::raw('COUNT(p.id) as product_count'),
                DB::raw('SUM(p.stock) as total_quantity'),
                DB::raw('SUM(p.stock * p.purchase_price) as category_value')
            )
            ->groupBy('c.id', 'c.name')
            ->orderBy('category_value', 'desc')
            ->get();

        $totalProducts = $currentInventory->count();
        $totalInventoryValue = $currentInventory->sum('inventory_value');
        $lowStockCount = $lowStockProducts->count();
        $estimatedReorderCost = $lowStockProducts->sum('estimated_cost');

        return view('', compact(
            'currentInventory', 'lowStockProducts', 'inventoryByCategory',
            'totalProducts', 'totalInventoryValue', 'lowStockCount', 'estimatedReorderCost'
        ));
    }
```

- Laporan Inventory dan Stok (inventoryReport)
  Laporan ini mencakup: Nilai total inventaris saat ini, Daftar produk dengan stok rendah, Jumlah total produk dan nilai per kategori



```php
public function productProfitabilityReport(Request $request)
    {
        $startDate = $request->input('start_date', Carbon::now()->startOfMonth()->format('Y-m-d'));
        $endDate = $request->input('end_date', Carbon::now()->endOfMonth()->format('Y-m-d'));

        $productProfitability = DB::table('products as p')
            ->leftJoin('sale_details as sd', 'p.id', '=', 'sd.product_id')
            ->leftJoin('sales as s', 'sd.sale_id', '=', 's.id')
            ->join('categories as c', 'p.category_id', '=', 'c.id')
            ->select(
                'c.name as category',
                'p.name as product_name',
                'p.code',
                'p.purchase_price',
                'p.selling_price',
                DB::raw('p.selling_price - p.purchase_price as margin_amount'),
                DB::raw('ROUND(((p.selling_price - p.purchase_price) / p.purchase_price) * 100, 2) as margin_percent'),
                DB::raw('COALESCE(SUM(CASE WHEN s.sale_date BETWEEN ? AND ? THEN sd.quantity ELSE 0 END), 0) as total_sold'),
                DB::raw('COALESCE(SUM(CASE WHEN s.sale_date BETWEEN ? AND ? THEN sd.subtotal ELSE 0 END), 0) as total_revenue'),
                DB::raw('COALESCE(SUM(CASE WHEN s.sale_date BETWEEN ? AND ? THEN (sd.selling_price - p.purchase_price) * sd.quantity ELSE 0 END), 0) as total_profit'),
                'p.stock as current_stock'
            )
            ->groupBy('c.name', 'p.id', 'p.name', 'p.code', 'p.purchase_price', 'p.selling_price', 'p.stock')
            ->orderBy('total_profit', 'desc')
            ->setBindings([$startDate, $endDate, $startDate, $endDate, $startDate, $endDate])
            ->get();

        // Best performing products
        $bestPerformers = $productProfitability->where('total_sold', '>', 0)->take(10);

        // Worst performing products (yang ada stok tapi tidak laku)
        $worstPerformers = $productProfitability
            ->where('current_stock', '>', 0)
            ->where('total_sold', 0)
            ->take(10);

        // Profitability by category
        $categoryProfitability = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales a
```

- Fungsi ini Menyajikan data margin keuntungan per produk, baik dalam nilai maupun persen. Produk dengan performa terbaik dan terburuk dipisahkan untuk membantu pengambilan keputusan. Juga dihitung profitabilitas per kategori dan produk yang belum laku tetapi masih memiliki stok. Hasil analisis ini tampil dalam view laporan-Inventory.



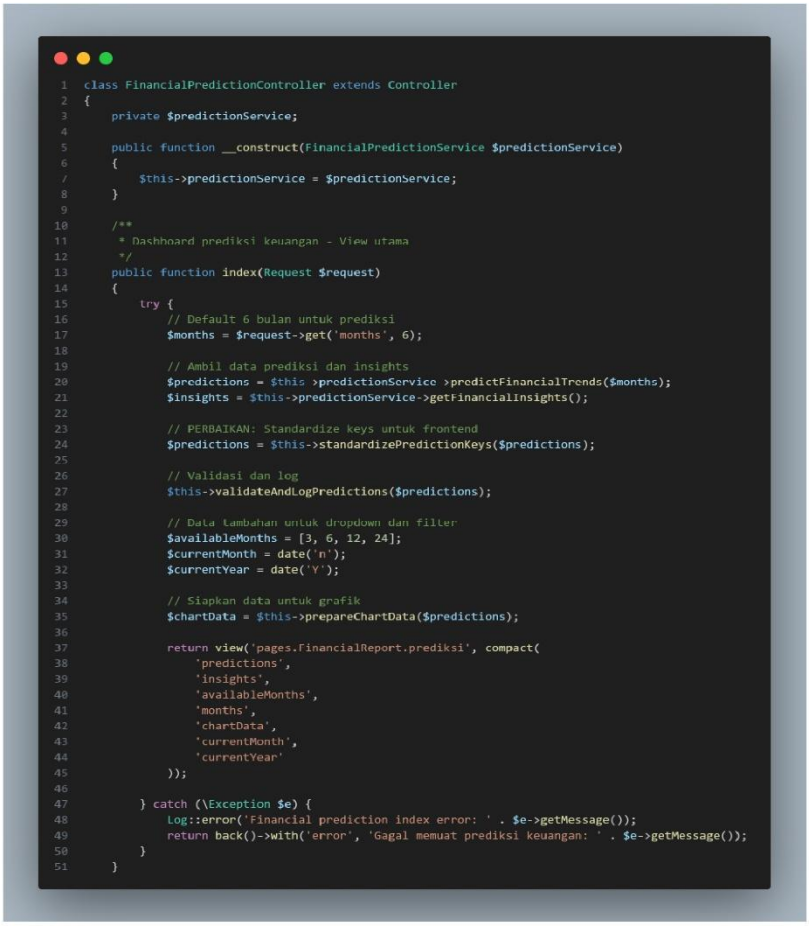
```php
public function dashboardOverview(Request $request)
    {
        $startDate = $request->input('start_date', Carbon::now()->startOfMonth()->format('Y-m-d'));
        $endDate = $request->input('end_date', Carbon::now()->endOfMonth()->format('Y-m-d'));

        // Key metrics
        $totalRevenue = DB::table('sales')
            ->whereBetween('sale_date', [$startDate, $endDate])
            ->sum('final_amount');

        $totalCogs = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales as s', 'sd.sale_id', '=', 's.id')
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->sum(DB::raw('sd.quantity * p.purchase_price'));

        $totalTransactions = DB::table('sales')
            ->whereBetween('sale_date', [$startDate, $endDate])
            ->count();

        $lowStockCount = DB::table('products')
            ->where('stock', '<=', DB::raw('min_stock'))
            ->count();

        $totalInventoryValue = DB::table('products')
            ->sum(DB::raw('stock * purchase_price'));

        $grossProfit = $totalRevenue - $totalCogs;
        $grossMarginPercent = $totalRevenue > 0 ? round(($grossProfit / $totalRevenue) * 100, 2) : 0;
        $averageTransaction = $totalTransactions > 0 ? round($totalRevenue / $totalTransactions, 2) : 0;

        return view('pages.FinancialReport.index', compact(
            'startDate', 'endDate', 'totalRevenue', 'totalCogs', 'grossProfit',
            'grossMarginPercent', 'totalTransactions', 'averageTransaction',
            'lowStockCount', 'totalInventoryValue'
        ));
    }
```

- Fungsi ini Menyediakan ikhtisar semua metrik utama: Pendapatan, HPP, laba kotor, Margin laba kotor dalam persen, Jumlah transaksi, transaksi rata-rata, Jumlah stok rendah & total nilai inventari

```php
public function monthlySummary(Request $request)
    {
        $year = $request->input('year', Carbon::now()->year);
        $month = $request->input('month', Carbon::now()->month);

        $startDate = Carbon::createFromDate($year, $month, 1)->startOfMonth()->format('Y-m-d');
        $endDate = Carbon::createFromDate($year, $month, 1)->endOfMonth()->format('Y-m-d');

        // Daily sales for the month
        $dailySales = DB::table('sales')
            ->select(
                DB::raw('DATE(sale_date) as date'),
                DB::raw('COUNT(*) as transaction_count'),
                DB::raw('SUM(final_amount) as daily_revenue')
            )
            ->whereBetween('sale_date', [$startDate, $endDate])
            ->groupBy(DB::raw('DATE(sale_date)'))
            ->orderBy('date')
            ->get();

        // Top selling products for the month
        $topProducts = DB::table('sale_details as sd')
            ->join('products as p', 'sd.product_id', '=', 'p.id')
            ->join('sales as s', 'sd.sale_id', '=', 's.id')
            ->select(
                'p.name as product_name',
                DB::raw('SUM(sd.quantity) as total_sold'),
                DB::raw('SUM(sd.subtotal) as total_revenue')
            )
            ->whereBetween('s.sale_date', [$startDate, $endDate])
            ->groupBy('p.id', 'p.name')
            ->orderBy('total_revenue', 'desc')
            ->limit(10)
            ->get();

        // Monthly totals
        $monthlyTotals = DB::table('sales')
            ->whereBetween('sale_date', [$startDate, $endDate])
            ->selectRaw('
                COUNT(*) as total_transactions,
                SUM(final_amount) as total_revenue,
                AVG(final_amount) as average_transaction
            ')
            ->first();

        $monthName = Carbon::createFromDate($year, $month, 1)->format('F Y');

        return view('pages.FinancialReport.Laporan-Ringkasan-Bulanan', compact(
            'year', 'month', 'monthName', 'startDate', 'endDate',
            'dailySales', 'topProd
```

- Fungsi ini Menampilkan ringkasan bulanan dalam bentuk: Produk terlaris bulan itu, Total transaksi dan pendapatan bulanan

FINANCIAL PREDICTIONS CONTROLLER



```php
class FinancialPredictionController extends Controller
{
    private $predictionService;

    public function __construct(FinancialPredictionService $predictionService)
    {
        $this->predictionService = $predictionService;
    }

    /**
     * Dashboard prediksi keuangan - View utama
     */
    public function index(Request $request)
    {
        try {
            // Default 6 bulan untuk prediksi
            $months = $request->get('months', 6);

            // Ambil data prediksi dan insights
            $predictions = $this->predictionService->predictFinancialTrends($months);
            $insights = $this->predictionService->getFinancialInsights();

            // PERBAIKAN: Standardize keys untuk frontend
            $predictions = $this->standardizePredictionKeys($predictions);

            // Validasi dan log
            $this->validateAndLogPredictions($predictions);

            // Data tambahan untuk dropdown dan filter
            $availableMonths = [3, 6, 12, 24];
            $currentMonth = date('n');
            $currentYear = date('Y');

            // Siapkan data untuk grafik
            $chartData = $this->prepareChartData($predictions);

            return view('pages.financialReport.prediksi', compact(
                'predictions',
                'insights',
                'availableMonths',
                'months',
                'chartData',
                'currentMonth',
                'currentYear'
            ));

        } catch (\Exception $e) {
            Log::error('Financial prediction index error: ' . $e->getMessage());
            return back()->with('error', 'Gagal memuat prediksi keuangan: ' . $e->getMessage());
        }
    }
```

- Fungsi ini bertanggung jawab menampilkan halaman utama prediksi keuangan. Setelah parameter jumlah bulan diambil dari permintaan (default 6 bulan), sistem memanggil service predictFinancialTrends() dan getFinancialInsights() untuk mendapatkan data tren dan insight keuangan. Data yang diperoleh kemudian disesuaikan key-nya menggunakan standardizePredictionKeys() untuk memastikan konsistensi dengan struktur frontend. Fungsi ini juga mempersiapkan data grafik menggunakan prepareChartData() dan mengirim seluruh data ke view pages.FinancialReport.prediksi

```php
public function getPredictions(Request $request): JsonResponse
    {
        try {
            $months = $request->get('months', 6);

            Log::info("Fetching predictions for {$months} months");

            $predictions = $this->predictionService->predictFinancialTrends($months);

            // PERBAIKAN: Standardize keys sebelum dikirim ke frontend
            $predictions = $this->standardizePredictionKeys($predictions);

            Log::info('Raw predictions from service:', $predictions);

            // Validasi data
            $this->validateAndLogPredictions($predictions);

            // Siapkan chart data
            $chartData = $this->prepareChartData($predictions);

            Log::info('Prepared chart data:', $chartData);

            // PERBAIKAN: Pastikan structure response sesuai ekspektasi frontend
            $response = [
                'success' => true,
                'data' => $predictions,
                'chartData' => $chartData
            ];

            Log::info('Final response structure:', [
                'has_data' => isset($response['data']),
                'has_summary' => isset($response['data']['summary']),
                'summary_keys' => isset($response['data']['summary']) ? array_keys($response['data']['summary']) : [],
                'has_chartData' => isset($response['chartData'])
            ]);

            return response()->json($response);

        } catch (\Exception $e) {
            Log::error('Get predictions error: ' . $e->getMessage());
            Log::error('Stack trace: ' . $e->getTraceAsString());

            return response()->json([
                'success' => false,
                'message' => 'Gagal memuat prediksi: ' . $e->getMessage(),
                'debug' => config('app.debug') ? $e->getTraceAsString() : null
            ], 500);
        }
    }
```

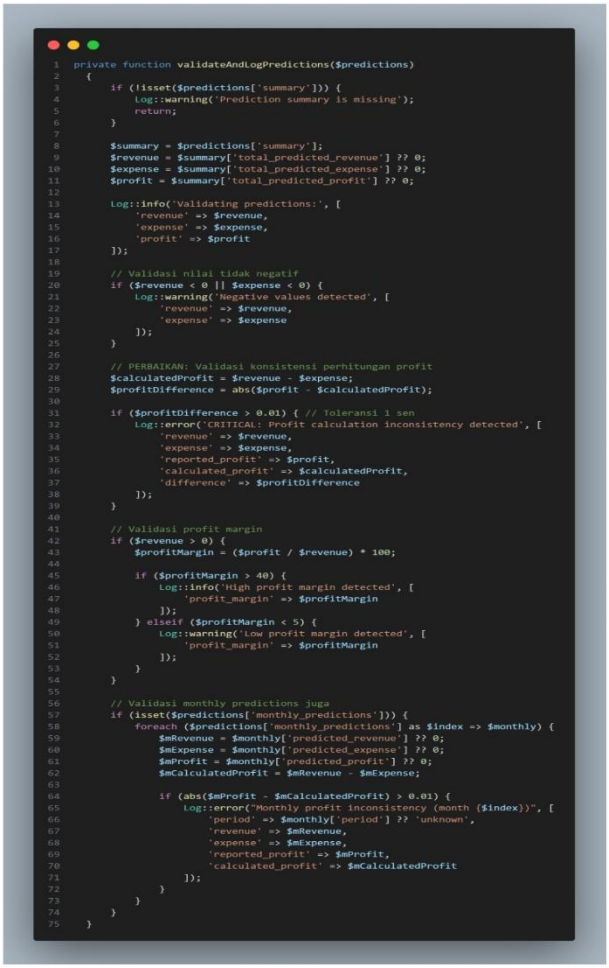
- Fungsi ini melayani permintaan AJAX frontend untuk memuat data prediksi. Setelah menentukan jumlah bulan prediksi, fungsi ini memanggil service untuk mengambil data tren, menstandarisasi key-nya, dan memvalidasi konsistensi datanya. Data prediksi dan grafik dikemas dalam format JSON yang sesuai ekspektasi frontend. Proses ini dilengkapi logging mendetail untuk memudahkan debugging saat terjadi ketidaksesuaian data.

```php
private function standardizePredictionKeys($predictions)
    {
        Log::info('Standardizing prediction keys...');
        Log::info('Original predictions structure:', array_keys($predictions));

        // Pastikan summary ada dan memiliki keys yang benar
        if (isset($predictions['summary'])) {
            $summary = $predictions['summary'];
            Log::info('Original summary keys:', array_keys($summary));

            // PERBAIKAN: Map semua kemungkinan variasi key ke standard
            $standardSummary = [];

            // Revenue mapping
            $standardSummary['total_predicted_revenue'] =
                $summary['total_predicted_revenue'] ??
                $summary['predicted_revenue'] ??
                $summary['revenue'] ?? 0;

            // PERBAIKAN UTAMA: Expense mapping - coba semua kemungkinan key
            $standardSummary['total_predicted_expense'] =
                $summary['total_predicted_expense'] ??
                $summary['total_predicted_expenses'] ??
                $summary['predicted_expense'] ??
                $summary['predicted_expenses'] ??
                $summary['expense'] ??
                $summary['expenses'] ?? 0;

            // PERBAIKAN KRITIS: Pastikan profit dihitung ulang dengan benar
            $revenue = (float) $standardSummary['total_predicted_revenue'];
            $expense = (float) $standardSummary['total_predicted_expense'];

            // Selalu hitung ulang profit untuk memastikan konsistensi
            $standardSummary['total_predicted_profit'] = $revenue - $expense;

            // Jika ada profit dari service, bandingkan dan log perbedaan
            $serviceProfitKeys = [
                'total_predicted_profit',
                'predicted_profit',
                'profit'
            ];

            $serviceProfit = null;
            foreach ($serviceProfitKeys as $key) {
                if (isset($summary[$key])) {
                    $serviceProfit = (float) $summary[$key];
                    break;
                }
            }

            if ($serviceProfit !== null) {
                $calculatedProfit = $standardSummary['total_predicted_profit'];
                $difference = abs($serviceProfit - $calculatedProfit);

                if ($difference > 0.01) { // Toleransi 1 sen
                    Log::warning('Profit calculation mismatch detected', [
                        'service_profit' => $serviceProfit,
                        'calculated_profit' => $calculatedProfit,
                        'revenue' => $revenue,
                        'expense' => $expense,
                        'difference' => $difference
                    ]);
                }
            }

            // Pastikan semua nilai adalah float
            $standardSummary['total_predicted_revenue'] = $revenue;
            $standardSummary['total_predicted_expense'] = $expense;
            $standardSummary['total_predicted_profit'] = $standardSummary['total_predicted_profit'];

            $predictions['summary'] = $standardSummary;

            Log::info('Standardized summary:', $standardSummary);
        } else {
            Log::warning('No summary found in predictions');

            // Buat summary default jika tidak ada
            $predictions['summary'] = [
                'total_predicted_revenue' => 0.0,
                'total_predicted_expense' => 0.0,
                'total_predicted_profit' => 0.0
            ];
        }

        // PERBAIKAN KRITIS: Standardize monthly predictions dengan perhitungan ulang profit
        if (isset($predictions['monthly_predictions']) && is_array($predictions['monthly_predictions'])) {
            foreach ($predictions['monthly_predictions'] as $index => &$monthlyPred) {
                // Ambil dan standardize revenue
                $revenue = (float) ($monthlyPred['predicted_revenue'] ??
                                    $monthlyPred['revenue'] ?? 0);

                // Ambil dan standardize expense dengan berbagai kemungkinan key
                $expense = (float) ($monthlyPred['predicted_expense'] ??
                                    $monthlyPred['predicted_expenses'] ??
                                    $monthlyPred['expense'] ??
                                    $monthlyPred['expenses'] ?? 0);

                // SELALU hitung ulang profit untuk memastikan konsistensi
                $calculatedProfit = $revenue - $expense;

                // Ambil profit dari service jika ada untuk perbandingan
                $serviceProfit = (float) ($monthlyPred['predicted_profit'] ??
                                          $monthlyPred['profit'] ?? null);

                // Log jika ada perbedaan dengan service profit
                if ($serviceProfit !== null && abs($serviceProfit - $calculatedProfit) > 0.01) {
                    Log::warning("Monthly prediction profit mismatch (month {$index})", [
                        'service_profit' => $serviceProfit,
                        'calculated_profit' => $calculatedProfit,
                        'revenue' => $revenue,
                        'expense' => $expense,
                        'period' => $monthlyPred['period'] ?? 'unknown'
                    ]);
                }

                // Set nilai yang sudah distandarisasi
                $monthlyPred['predicted_revenue'] = $revenue;
                $monthlyPred['predicted_expense'] = $expense;
                $monthlyPred['predicted_profit'] = $calculatedProfit; // Gunakan hasil perhitungan

                // Validasi nilai tidak boleh negatif untuk revenue dan expense
                if ($revenue < 0 || $expense < 0) {
                    Log::warning("Negative values in monthly prediction (month {$index})", [
                        'revenue' => $revenue,
                        'expense' => $expense,
                        'period' => $monthlyPred['period'] ?? 'unknown'
                    ]);
                }
            }
            unset($monthlyPred); // Hapus reference
        }

        // Pastikan accuracy ada
        if (!isset($predictions['accuracy'])) {
            $predictions['accuracy'] = 0.0;
        } else {
            $predictions['accuracy'] = (float) $predictions['accuracy'];
        }

        Log::info('Final standardized predictions structure:', [
            'summary_keys' => array_keys($predictions['summary']),
            'summary_values' => $predictions['summary'],
            'monthly_count' => count($predictions['monthly_predictions'] ?? []),
            'accuracy' => $predictions['accuracy'],
            'first_monthly_sample' => $predictions['monthly_predictions'][0] ?? null
        ]);

        return $predictions;
    }
```

- Fungsi standardizePredictionKeys()
  Fungsi ini melakukan normalisasi struktur key pada array summary dan monthly_predictions agar tetap konsisten meskipun terjadi perubahan struktur di service. Fungsi ini: Memastikan kunci revenue, expense, dan profit ada dan memiliki nilai numerik, Selalu menghitung ulang profit sebagai revenue – expense, Melakukan logging apabila terdapat ketidaksesuaian nilai profit dari service, Memastikan semua nilai bertipe float dan tidak negatif. Menangani berbagai variasi key dari service (revenue, predicted_revenue,

```php
public function exportPdf(Request $request)
    {
        try {
            $months = $request->get('months', 6);
            $predictions = $this->predictionService->predictFinancialTrends($months);
            $insights = $this->predictionService->getFinancialInsights();

            // Standardize untuk PDF juga
            $predictions = $this->standardizePredictionKeys($predictions);
            $this->validateAndLogPredictions($predictions);

            $pdf = app('dompdf.wrapper');
            $pdf->loadView('financial.predictions-pdf', compact('predictions', 'insights', 'months'));

            return $pdf->download('financial-predictions-' . date('Y-m-d') . '.pdf');
        } catch (\Exception $e) {
            Log::error('Export PDF error: ' . $e->getMessage());
            return back()->with('error', 'Gagal export PDF: ' . $e->getMessage());
        }
    }

```

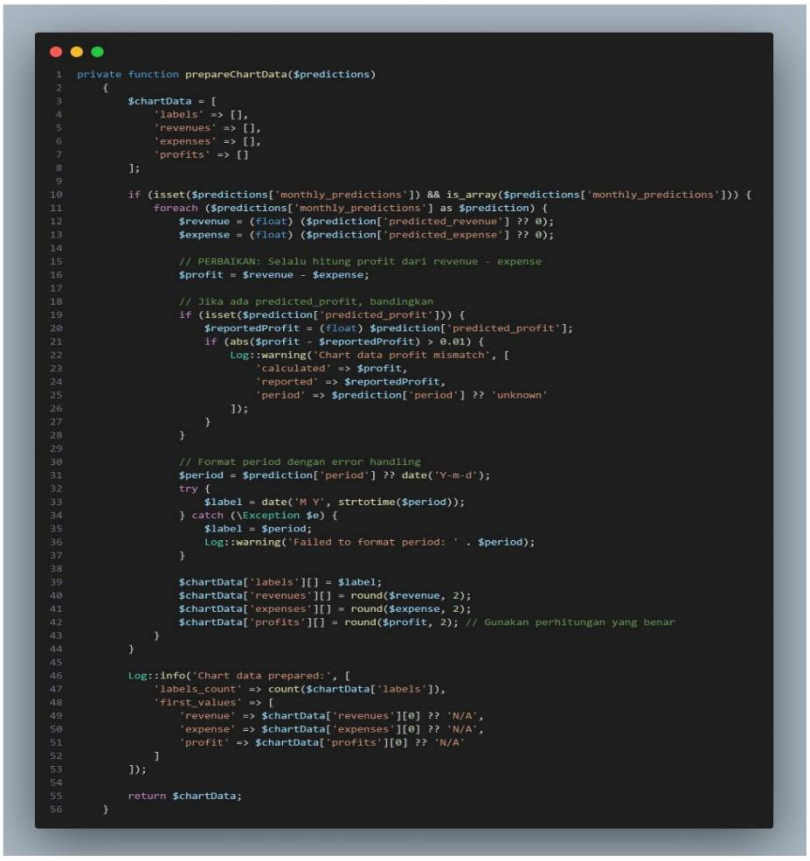
- Fungsi exportPdf()
  Fungsi ini mengekspor hasil prediksi dan insight keuangan ke dalam file PDF. Setelah data diproses dan distandarisasi, sistem merender view financial.predictions-pdf dan mengunduh hasilnya sebagai file PDF dengan nama dinamis berdasarkan tanggal saat ini.



```php
private function validateAndLogPredictions($predictions)
    {
        if (!isset($predictions['summary'])) {
            Log::warning('Prediction summary is missing');
            return;
        }

        $summary = $predictions['summary'];
        $revenue = $summary['total_predicted_revenue'] ?? 0;
        $expense = $summary['total_predicted_expense'] ?? 0;
        $profit = $summary['total_predicted_profit'] ?? 0;

        Log::info('Validating predictions:', [
            'revenue' => $revenue,
            'expense' => $expense,
            'profit' => $profit
        ]);

        // Validasi nilai tidak negatif
        if ($revenue < 0 || $expense < 0) {
            Log::warning('Negative values detected', [
                'revenue' => $revenue,
                'expense' => $expense
            ]);
        }

        // PERBAIKAN: Validasi konsistensi perhitungan profit
        $calculatedProfit = $revenue - $expense;
        $profitDifference = abs($profit - $calculatedProfit);

        if ($profitDifference > 0.01) { // Toleransi 1 sen
            Log::error('CRITICAL: Profit calculation inconsistency detected', [
                'revenue' => $revenue,
                'expense' => $expense,
                'reported_profit' => $profit,
                'calculated_profit' => $calculatedProfit,
                'difference' => $profitDifference
            ]);
        }

        // Validasi profit margin
        if ($revenue > 0) {
            $profitMargin = ($profit / $revenue) * 100;

            if ($profitMargin > 40) {
                Log::info('High profit margin detected', [
                    'profit_margin' => $profitMargin
                ]);
            } elseif ($profitMargin < 5) {
                Log::warning('Low profit margin detected', [
                    'profit_margin' => $profitMargin
                ]);
            }
        }

        // Validasi monthly predictions juga
        if (isset($predictions['monthly_predictions'])) {
            foreach ($predictions['monthly_predictions'] as $index => $monthly) {
                $mRevenue = $monthly['predicted_revenue'] ?? 0;
                $mExpense = $monthly['predicted_expense'] ?? 0;
                $mProfit = $monthly['predicted_profit'] ?? 0;
                $mCalculatedProfit = $mRevenue - $mExpense;

                if (abs($mProfit - $mCalculatedProfit) > 0.01) {
                    Log::error("Monthly profit inconsistency (month {$index})", [
                        'period' => $monthly['period'] ?? 'unknown',
                        'revenue' => $mRevenue,
                        'expense' => $mExpense,
                        'reported_profit' => $mProfit,
                        'calculated_profit' => $mCalculatedProfit
                    ]);
                }
            }
        }
    }
```

- Fungsi validateAndLogPredictions()
  Fungsi ini memeriksa konsistensi dan kewajaran data prediksi yang telah distandarisasi. Validasi yang dilakukan mencakup
  1. Pemeriksaan keberadaan summary
  2. Validasi nilai numerik, Mengecek apakah revenue dan expense bernilai negatif. Jika iya, log peringatan akan dicatat.
  3. Validasi konsistensi profit, Profit dihitung ulang sebagai revenue - expense. Jika selisih dengan nilai profit yang dilaporkan lebih dari 0.01, akan dicatat sebagai error log "CRITICAL".

4. Validasi margin keuntungan, Jika margin profit di atas 40% atau di bawah 5%, sistem mencatat log informasi atau peringatan.
5. Validasi prediksi bulanan, Untuk setiap bulan, dilakukan pengecekan konsistensi perhitungan revenue - expense = profit. Jika terjadi selisih yang signifikan, dicatat dalam log sebagai "Monthly profit inconsistency".



```php
private function prepareChartData($predictions)
    {
        $chartData = [
            'labels' => [],
            'revenues' => [],
            'expenses' => [],
            'profits' => []
        ];

        if (isset($predictions['monthly_predictions']) && is_array($predictions['monthly_predictions'])) {
            foreach ($predictions['monthly_predictions'] as $prediction) {
                $revenue = (float) ($prediction['predicted_revenue'] ?? 0);
                $expense = (float) ($prediction['predicted_expense'] ?? 0);

                // PERBAIKAN: Selalu hitung profit dari revenue - expense
                $profit = $revenue - $expense;

                // Jika ada predicted_profit, bandingkan
                if (isset($prediction['predicted_profit'])) {
                    $reportedProfit = (float) $prediction['predicted_profit'];
                    if (abs($profit - $reportedProfit) > 0.01) {
                        Log::warning('Chart data profit mismatch', [
                            'calculated' => $profit,
                            'reported' => $reportedProfit,
                            'period' => $prediction['period'] ?? 'unknown'
                        ]);
                    }
                }

                // Format period dengan error handling
                $period = $prediction['period'] ?? date('Y-m-d');
                try {
                    $label = date('M Y', strtotime($period));
                } catch (\Exception $e) {
                    $label = $period;
                    Log::warning('Failed to format period: ' . $period);
                }

                $chartData['labels'][] = $label;
                $chartData['revenues'][] = round($revenue, 2);
                $chartData['expenses'][] = round($expense, 2);
                $chartData['profits'][] = round($profit, 2); // Gunakan perhitungan yang benar
            }
        }

        Log::info('Chart data prepared:', [
            'labels_count' => count($chartData['labels']),
            'first_values' => [
                'revenue' => $chartData['revenues'][0] ?? 'N/A',
                'expense' => $chartData['expenses'][0] ?? 'N/A',
                'profit' => $chartData['profits'][0] ?? 'N/A'
            ]
        ]);

        return $chartData;
    }
```

- Fungsi ini menyusun data prediksi bulanan ke dalam struktur yang dibutuhkan oleh grafik di frontend. Data diformat dalam array label (bulan), revenue, expense, dan profit. Profit selalu dihitung ulang agar valid, dan setiap ketidaksesuaian dengan nilai yang dilaporkan akan dicatat dalam log untuk debugging.

PRODUK ANALISIS CONTROLLER

```
1  public function index()
2      {
3          return view('pages.produk.analisis');
4      }
5
```

- Fungsi index()
  Fungsi ini menampilkan halaman awal untuk analisis produk, tanpa menjalankan proses analisis apa pun. Ini hanya merender tampilan pages.produk.analisis.

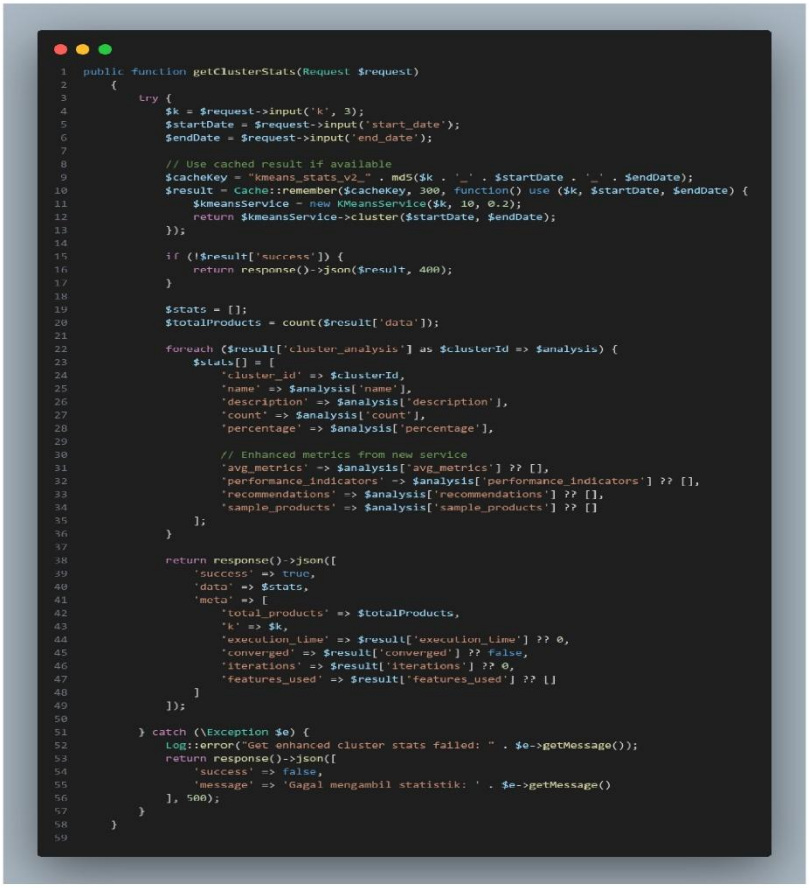
```php
public function analyze(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'start_date' => 'nullable|date|before_or_equal:today',
            'end_date' => 'nullable|date|after_or_equal:start_date|before_or_equal:today',
            'k' => 'nullable|integer|min:2|max:8'
        ]);

        if ($validator->fails()) {
            return response()->json([
                'success' => false,
                'message' => 'Validation failed',
                'errors' => $validator->errors()
            ], 422);
        }

        try {
            $k = $request->input('k', 3);
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');

            // Validate date range (not more than 1 year)
            if ($startDate && $endDate) {
                $start = \Carbon\Carbon::parse($startDate);
                $end = \Carbon\Carbon::parse($endDate);

                if ($end->diffInDays($start) > 365) {
                    return response()->json([
                        'success' => false,
                        'message' => 'Rentang waktu tidak boleh lebih dari 1 tahun'
                    ], 422);
                }
            }

            // Check if analysis is already running
            $lockKey = 'kmeans_running_' . md5($k . '_' . $startDate . '_' . $endDate);
            if (Cache::has($lockKey)) {
                return response()->json([
                    'success' => false,
                    'message' => 'Analisis sedang berjalan. Silakan tunggu beberapa saat.',
                    'retry_after' => Cache::get($lockKey . '_start_time') ?
                        max(0, 300 - (time() - Cache::get($lockKey . '_start_time'))) : 300
                ], 429);
            }

            // Set lock to prevent concurrent analysis
            Cache::put($lockKey, true, 300); // 5 minutes lock
            Cache::put($lockKey . '_start_time', time(), 300);

            Log::info("Starting enhanced K-Means analysis", [
                'k' => $k,
                'start_date' => $startDate,
                'end_date' => $endDate,
                'user_id' => auth()->id() ?? 'anonymous'
            ]);

            // Use optimized service with appropriate parameters based on K
            $maxIterations = min(20, 15 + $k); // More iterations for higher K
            $tolerance = max(0.1, 0.2 - ($k * 0.02)); // Tighter tolerance for higher K

            $kmeansService = new KMeansService($k, $maxIterations, $tolerance);
            $result = $kmeansService->cluster($startDate, $endDate);

            // Remove lock
            Cache::forget($lockKey);
            Cache::forget($lockKey . '_start_time');

            if (!$result['success']) {
                Log::warning("Enhanced K-Means analysis failed", [
                    'message' => $result['message'],
                    'k' => $k,
                    'start_date' => $startDate,
                    'end_date' => $endDate
                ]);

                return response()->json($result, 400);
            }

            // Log success
            Log::info("Enhanced K-Means analysis completed successfully", [
                'k' => $k,
                'total_products' => $result['total_products'],
                'execution_time' => $result['execution_time'],
                'iterations' => $result['iterations'],
                'converged' => $result['converged']
            ]);

            return response()->json([
                'success' => true,
                'message' => 'Analisis berhasil dilakukan',
                'data' => [
                    'clusters' => $result['data'],
                    'analysis' => $result['cluster_analysis'],
                    'meta' => [
                        'iterations' => $result['iterations'],
                        'converged' => $result['converged'],
                        'execution_time' => $result['execution_time'],
                        'k' => $k,
                        'total_products' => $result['total_products'],
                        'features_used' => $result['features_used'],
                        'start_date' => $startDate,
                        'end_date' => $endDate,
                        'analysis_timestamp' => now()->toISOString()
                    ]
                ]
            ]);

        } catch (\Exception $e) {
            // Remove lock on error
            $lockKey = 'kmeans_running_' . md5($k . '_' . $startDate . '_' . $endDate);
            Cache::forget($lockKey);
            Cache::forget($lockKey . '_start_time');

            Log::error("Enhanced K-Means analysis failed", [
                'error' => $e->getMessage(),
                'file' => $e->getFile(),
                'line' => $e->getLine(),
                'k' => $k ?? null,
                'start_date' => $startDate ?? null,
                'end_date' => $endDate ?? null,
                'user_id' => auth()->id() ?? 'anonymous'
            ]);

            return response()->json([
                'success' => false,
                'message' => 'Terjadi kesalahan saat melakukan analisis: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi analyze()
  Fungsi ini menjalankan analisis klasterisasi produk menggunakan algoritma K-Means. Pertama, sistem memvalidasi input tanggal dan jumlah klaster (k). Jika input valid, maka sistem memastikan tidak ada analisis lain yang sedang berjalan dengan memanfaatkan cache kunci (lockKey). Setelah itu, sistem mengatur parameter iterasi dan toleransi berdasarkan nilai k, lalu memanggil service KMeansService untuk melakukan proses clustering. Jika berhasil, hasil analisis dikembalikan ke frontend dalam format JSON, lengkap dengan data klaster, insight, dan metainformasi seperti jumlah iterasi, waktu eksekusi, dan jumlah produk yang dianalisis. Jika gagal, sistem menghapus kunci cache dan mengembalikan pesan error.

```php
public function getAnalysisStatus(Request $request)
    {
        $k = $request->input('k', 3);
        $startDate = $request->input('start_date');
        $endDate = $request->input('end_date');

        $lockKey = "kmeans_running_" . md5($k . '_' . $startDate . '_' . $endDate);
        $isRunning = Cache::has($lockKey);

        $response = [
            'running' => $isRunning,
            'message' => $isRunning ? 'Analisis sedang berjalan...' : 'Tidak ada analisis yang berjalan'
        ];

        if ($isRunning) {
            $startTime = Cache::get($lockKey . '_start_time');
            if ($startTime) {
                $elapsed = time() - $startTime;
                $response['elapsed_time'] = $elapsed;
                $response['estimated_remaining'] = max(0, 300 - $elapsed); // Max 5 minutes
            }
        }

        return response()->json($response);
    }

```

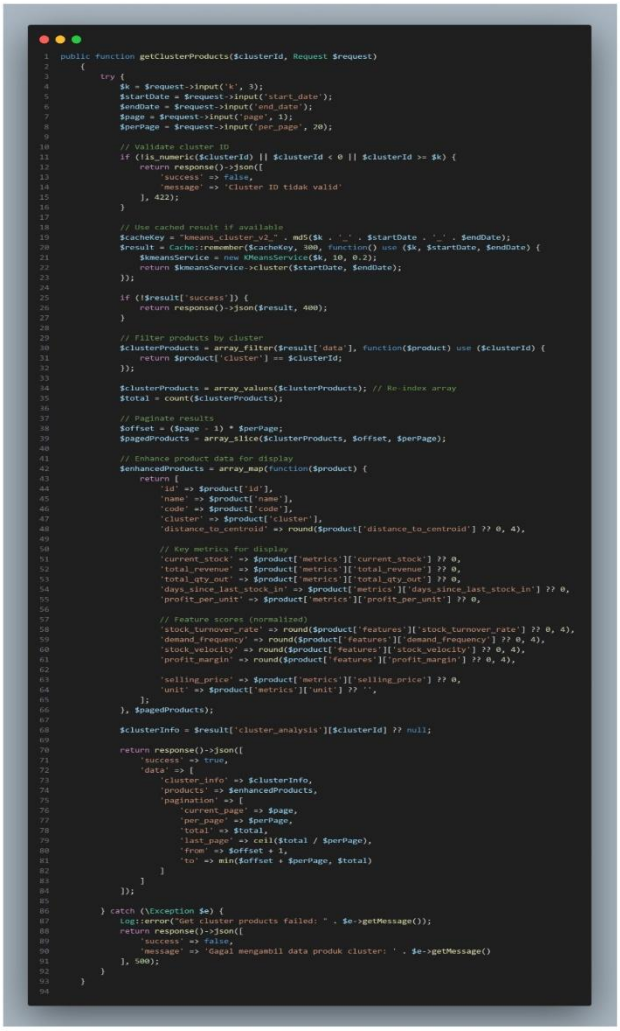
- Fungsi ini mengecek status analisis K-Means apakah sedang berjalan atau tidak, menggunakan informasi dari cache. Jika sedang berlangsung, fungsi ini juga mengembalikan estimasi waktu yang tersisa.

- Fungsi export()
  Fungsi ini mengekspor hasil analisis ke dalam format CSV atau JSON. Sistem mengambil data dari cache (atau menjalankan ulang klasterisasi jika belum ada cache), kemudian membentuk array berisi informasi produk dan metrik performa seperti stock_turnover_rate, demand_frequency, profit_margin, dan sebagainya. Jika format yang dipilih adalah csv, maka data dikonversi menggunakan fungsi arrayToCsv() dan dikirim sebagai file unduhan.



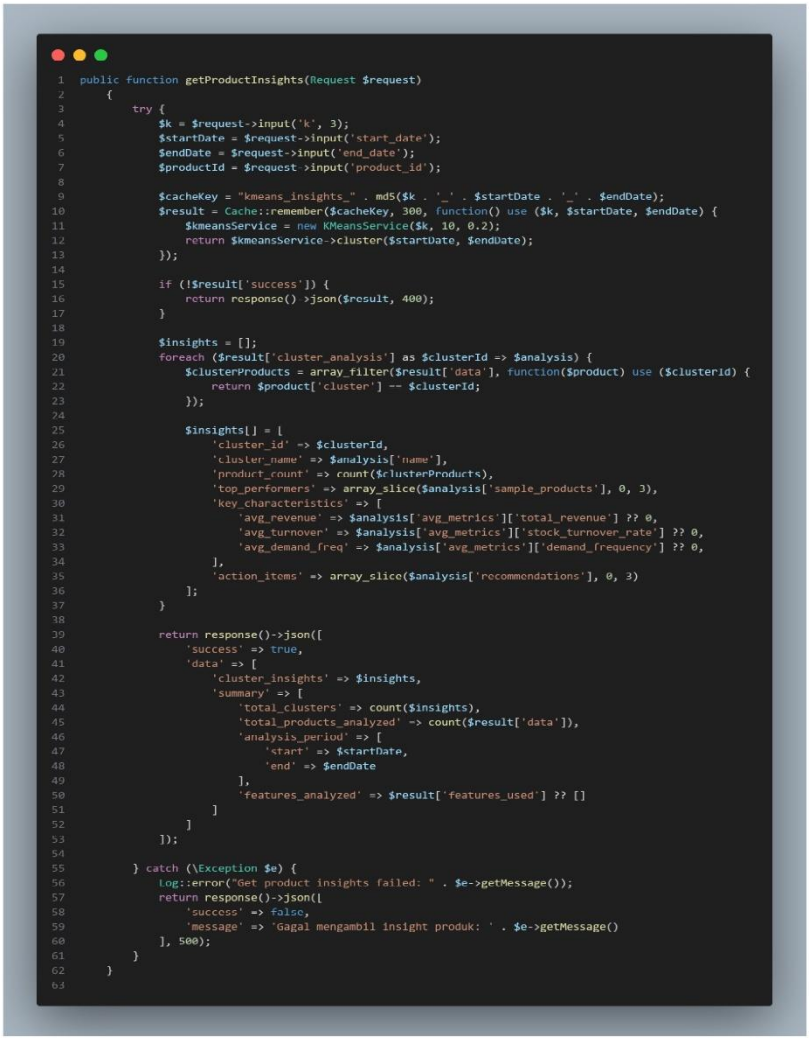
```php
public function getClusterStats(Request $request)
    {
        try {
            $k = $request->input('k', 3);
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');

            // Use cached result if available
            $cacheKey = "kmeans_stats_v2_" . md5($k . '_' . $startDate . '_' . $endDate);
            $result = Cache::remember($cacheKey, 300, function() use ($k, $startDate, $endDate) {
                $kmeansService = new KMeansService($k, 10, 0.2);
                return $kmeansService->cluster($startDate, $endDate);
            });

            if (!$result['success']) {
                return response()->json($result, 400);
            }

            $stats = [];
            $totalProducts = count($result['data']);

            foreach ($result['cluster_analysis'] as $clusterId => $analysis) {
                $stats[] = [
                    'cluster_id' => $clusterId,
                    'name' => $analysis['name'],
                    'description' => $analysis['description'],
                    'count' => $analysis['count'],
                    'percentage' => $analysis['percentage'],

                    // Enhanced metrics from new service
                    'avg_metrics' => $analysis['avg_metrics'] ?? [],
                    'performance_indicators' => $analysis['performance_indicators'] ?? [],
                    'recommendations' => $analysis['recommendations'] ?? [],
                    'sample_products' => $analysis['sample_products'] ?? []
                ];
            }

            return response()->json([
                'success' => true,
                'data' => $stats,
                'meta' => [
                    'total_products' => $totalProducts,
                    'k' => $k,
                    'execution_time' => $result['execution_time'] ?? 0,
                    'converged' => $result['converged'] ?? false,
                    'iterations' => $result['iterations'] ?? 0,
                    'features_used' => $result['features_used'] ?? []
                ]
            ]);

        } catch (\Exception $e) {
            Log::error('Get enhanced cluster stats failed: ' . $e->getMessage());
            return response()->json([
                'success' => false,
                'message' => 'Gagal mengambil statistik: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi getClusterStats()
  Fungsi ini mengambil statistik ringkasan dari hasil klasterisasi, seperti jumlah produk per klaster, deskripsi performa, dan metrik rata-rata. Data diambil dari cache (jika tersedia) atau dilakukan klasterisasi ulang jika belum ada. Fungsi ini berguna untuk menampilkan ringkasan performa tiap klaster di frontend



```php
public function getClusterProducts($clusterId, Request $request)
    {
        try {
            $k = $request->input('k', 3);
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');
            $page = $request->input('page', 1);
            $perPage = $request->input('per_page', 20);

            // Validate cluster ID
            if (!is_numeric($clusterId) || $clusterId < 0 || $clusterId >= $k) {
                return response()->json([
                    'success' => false,
                    'message' => 'Cluster ID tidak valid'
                ], 422);
            }

            // Use cached result if available
            $cacheKey = "kmeans_cluster_v2_" . md5($k . '_' . $startDate . '_' . $endDate);
            $result = Cache::remember($cacheKey, 300, function() use ($k, $startDate, $endDate) {
                $kmeansService = new KMeansService($k, 10, 0.2);
                return $kmeansService->cluster($startDate, $endDate);
            });

            if (!$result['success']) {
                return response()->json($result, 400);
            }

            // Filter products by cluster
            $clusterProducts = array_filter($result['data'], function($product) use ($clusterId) {
                return $product['cluster'] == $clusterId;
            });

            $clusterProducts = array_values($clusterProducts); // Re-index array
            $total = count($clusterProducts);

            // Paginate results
            $offset = ($page - 1) * $perPage;
            $pagedProducts = array_slice($clusterProducts, $offset, $perPage);

            // Enhance product data for display
            $enhancedProducts = array_map(function($product) {
                return [
                    'id' => $product['id'],
                    'name' => $product['name'],
                    'code' => $product['code'],
                    'cluster' => $product['cluster'],
                    'distance_to_centroid' => round($product['distance_to_centroid'] ?? 0, 4),

                    // Key metrics for display
                    'current_stock' => $product['metrics']['current_stock'] ?? 0,
                    'total_revenue' => $product['metrics']['total_revenue'] ?? 0,
                    'total_qty_out' => $product['metrics']['total_qty_out'] ?? 0,
                    'days_since_last_stock_in' => $product['metrics']['days_since_last_stock_in'] ?? 0,
                    'profit_per_unit' => $product['metrics']['profit_per_unit'] ?? 0,

                    // Feature scores (normalized)
                    'stock_turnover_rate' => round($product['features']['stock_turnover_rate'] ?? 0, 4),
                    'demand_frequency' => round($product['features']['demand_frequency'] ?? 0, 4),
                    'stock_velocity' => round($product['features']['stock_velocity'] ?? 0, 4),
                    'profit_margin' => round($product['features']['profit_margin'] ?? 0, 4),

                    'selling_price' => $product['metrics']['selling_price'] ?? 0,
                    'unit' => $product['metrics']['unit'] ?? '',
                ];
            }, $pagedProducts);

            $clusterInfo = $result['cluster_analysis'][$clusterId] ?? null;

            return response()->json([
                'success' => true,
                'data' => [
                    'cluster_info' => $clusterInfo,
                    'products' => $enhancedProducts,
                    'pagination' => [
                        'current_page' => $page,
                        'per_page' => $perPage,
                        'total' => $total,
                        'last_page' => ceil($total / $perPage),
                        'from' => $offset + 1,
                        'to' => min($offset + $perPage, $total)
                    ]
                ]
            ]);

        } catch (\Exception $e) {
            Log::error("Get cluster products failed: " . $e->getMessage());
            return response()->json([
                'success' => false,
                'message' => 'Gagal mengambil data produk cluster: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi getClusterProducts()
  Fungsi ini menampilkan daftar produk berdasarkan klaster tertentu. Sistem mengambil hasil klasterisasi dari cache, lalu memfilter produk yang termasuk dalam ID klaster yang diminta. Data dipaginasi dan setiap produk diperkaya dengan metrik performa, jarak ke centroid, dan informasi stok.



```php
public function getProductInsights(Request $request)
    {
        try {
            $k = $request->input('k', 3);
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');
            $productId = $request->input('product_id');

            $cacheKey = "kmeans_insights_" . md5($k . '_' . $startDate . '_' . $endDate);
            $result = Cache::remember($cacheKey, 300, function() use ($k, $startDate, $endDate) {
                $kmeansService = new KMeansService($k, 10, 0.2);
                return $kmeansService->cluster($startDate, $endDate);
            });

            if (!$result['success']) {
                return response() >json($result, 400);
            }

            $insights = [];
            foreach ($result['cluster_analysis'] as $clusterId => $analysis) {
                $clusterProducts = array_filter($result['data'], function($product) use ($clusterId) {
                    return $product['cluster'] -- $clusterId;
                });

                $insights[] = [
                    'cluster_id' => $clusterId,
                    'cluster_name' => $analysis['name'],
                    'product_count' => count($clusterProducts),
                    'top_performers' => array_slice($analysis['sample_products'], 0, 3),
                    'key_characteristics' => [
                        'avg_revenue' => $analysis['avg_metrics']['total_revenue'] ?? 0,
                        'avg_turnover' => $analysis['avg_metrics']['stock_turnover_rate'] ?? 0,
                        'avg_demand_freq' => $analysis['avg_metrics']['demand_frequency'] ?? 0,
                    ],
                    'action_items' => array_slice($analysis['recommendations'], 0, 3)
                ];
            }

            return response()->json([
                'success' => true,
                'data' => [
                    'cluster_insights' => $insights,
                    'summary' => [
                        'total_clusters' => count($insights),
                        'total_products_analyzed' -> count($result['data']),
                        'analysis_period' => [
                            'start' => $startDate,
                            'end' => $endDate
                        ],
                        'features_analyzed' => $result['features_used'] ?? []
                    ]
                ]
            ]);

        } catch (\Exception $e) {
            Log::error("Get product insights failed: " . $e->getMessage());
            return response()->json([
                'success' => false,
                'message' => 'Gagal mengambil insight produk: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi getProductInsights()
  Fungsi ini menyusun insight analisis untuk setiap klaster, seperti performer terbaik, karakteristik utama, dan saran tindakan (recommendation). Informasi ini disusun dalam format JSON dan digunakan untuk menampilkan ringkasan performa produk berdasarkan hasil klasterisasi.

```php
public function getAnalysisHistory(Request $request)
    {
        try {
            $k = $request->input('k', 3);
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');

            $cacheKey = "kmeans_features_v2_{$startDate}_{$endDate}_{$k}";
            $hasCachedData = Cache::has($cacheKey);

            return response()->json([
                'success' => true,
                'data' => [
                    'has_cached_data' => $hasCachedData,
                    'cache_key' => $cacheKey,
                    'parameters' => [
                        'k' => $k,
                        'start_date' => $startDate,
                        'end_date' => $endDate
                    ]
                ]
            ]);

        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Gagal mengambil riwayat analisis: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi getAnalysisHistory(Request $request)
  Fungsi ini memeriksa apakah cache hasil analisis dengan parameter tertentu masih tersedia. Ini digunakan untuk menginformasikan frontend agar tidak perlu melakukan analisis ulang jika data sudah tersedia.

```php
public function clearCache(Request $request)
    {
        try {
            $k = $request->input('k');
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');

            if ($k && $startDate && $endDate) {
                // Clear specific cache - updated cache keys
                $cacheKeys = [
                    "kmeans_features_v2_{$startDate}_{$endDate}_{$k}",
                    "kmeans_export_v2_" . md5($k . '_' . $startDate . '_' . $endDate),
                    "kmeans_stats_v2_" . md5($k . '_' . $startDate . '_' . $endDate),
                    "kmeans_cluster_v2_" . md5($k . '_' . $startDate . '_' . $endDate),
                    "kmeans_insights_" . md5($k . '_' . $startDate . '_' . $endDate),
                ];

                foreach ($cacheKeys as $key) {
                    Cache::forget($key);
                }

                return response()->json([
                    'success' => true,
                    'message' => 'Cache enhanced K-Means berhasil dihapus untuk parameter tersebut'
                ]);
            } else {
                return response()->json([
                    'success' => true,
                    'message' => 'Silakan spesifikasi parameter untuk menghapus cache'
                ]);
            }

        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Gagal menghapus cache: ' . $e->getMessage()
            ], 500);
        }
    }
```

- Fungsi clearCache(Request $request)
  Fungsi ini menghapus cache hasil analisis K-Means yang disimpan sebelumnya, berdasarkan parameter k, start_date, dan end_date. Ini berguna untuk membersihkan data lama atau memperbarui analisis.

Lampiran 5. Sertifikat HKI

REPUBLIK INDONESIA
KEMENTERIAN HUKUM

SURAT PENCATATAN CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC002025085545, 9 Juli 2025
Pencipta		
Nama	:	**Indra Kusuma, Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom. dkk**
Alamat	:	Desa Sigentong RT01 RW 04 Kecamatan Wanasari Kabupaten Brebes Jawa Tengah, Wanasari, Kab. Brebes, Jawa Tengah, 52252
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Pusat Penelitian dan Pengabdian Masyarakat (P3M) Politeknik Harapan Bersama**
Alamat	:	Jalan Mataram No. 9, Pesurungan Lor, Kecamatan Margadana, Margadana, Kota Tegal, Jawa Tengah, 52142
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**APLIKASI PENJUALAN DENGAN OPTIMALISASI STOK BARANG MENGGUNAKAN K-MEANS CLUSTERING DAN INTEGRASI WHATSAPP BOT PADA RAKA STORE**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	8 Juli 2025, di Kota Tegal
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor Pencatatan	:	000925806

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL
u.b
Direktur Hak Cipta dan Desain Industri

Agung Damarsasongko,SH.,MH.
NIP. 196912261994031001

Balai Besar Sertifikasi Elektronik

Disclaimer:
1. Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.
2. Surat Pencatatan ini telah disegel secara elektronik menggunakan segel elektronik yang ditertibkan oleh Balai Besar Sertifikasi Elektronik, Badan Siber dan Sandi Negara.
3. Surat Pencatatan ini dapat dibuktikan keasliannya dengan memindai kode QR pada dokumen ini dan informasi akan ditampilkan dalam browser.

Lampiran 6. Lembar Bimbingan

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN SKRIPSI

Nama	:	**Indra Kusuma**
NIM	:	**21090042**
No. Ponsel	:	**089603636937**
Judul TA	:	**APLIKASI PENJUALAN DENGAN OPTIMALISASI STOK BARANG MENGGUNAKAN *K-MEANS CLUSTERING* DAN INTEGRASI *WHATSAPP BOT* PADA RAKA *STORE***

Dosen Pembimbing I : **Ir. Ginanjar Wiro Sasmito, M.Kom., IPM., ASEAN Eng.**

No	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
	22/05/25	proposal	Revisi	
	18/06/25	proposal	perbaiki K-means clust	
	23/06/25	proposal	Oke	
	11/7/25	Laporan	Revisi Bab I	
	14/7/25	Laporan	Revisi Bab I & Bab II	

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

Tegal,
Dosen Pembimbing I

Ir. Gihanjar Wiro Sasmito, M.Kom., IPM., ASEAN Eng.
NIPY. 10.007.032

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN SKRIPSI

Nama	:	**Indra Kusuma**
NIM	:	**21090042**
No. Ponsel	:	**089603636937**
Judul TA	:	**APLIKASI PENJUALAN DENGAN OPTIMALISASI STOK BARANG MENGGUNAKAN *K-MEANS CLUSTERING* DAN INTEGRASI *WHATSAPP BOT* PADA RAKA *STORE***

Dosen Pembimbing II : **Sharfina Febbi Handayani, S.Kom., M.Kom.**

No	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1.	13 Maret 2025	Pengecekan Proposal	Latar belakang ditambahkan dan cek lagi rumusan masalah	sharfi.
2.	30 April 2025	Pengecekan alur Penelitian	poin penjelasan alur Penelitian disesuaikan dengan gambar	sharfi.

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

3.	21 Mei 2025	Pengecekan Progress Fitur Fungsional terkait aplikasi Raka Store	diperbaiki bagian Laporan dan kategori	Sharfu
4.	6 Juni 2025	Pengecekan Progress Fitur penjualan dan Implementasi K-Means	diperbaiki bagian Laporan pada aplikasi	Sharfu
5.	25 Juni 2025	Pengecekan Progress Fitur Laporan Raka Store	Lanjut untuk membuat manual book u/ website Raka Store	Sharfu.
6.	7 Juli 2025	Pengecekan manual book	Perbaiki pembuatan Laporan manual book dengan mention gambar sesuai laporan	Sharfu
7	14 Juli 2025	Laporan BAB 1 dan BAB II	Perbaiki penulisan gambar tambahkan highlight setelah Gap Analisis	Sharfu
8.	15 Juli 2025	Laporan BAB II dan BAB III	ACC Laporan dan direkomendasikan untuk sidang ujian hasil Skripsi	Sharfu

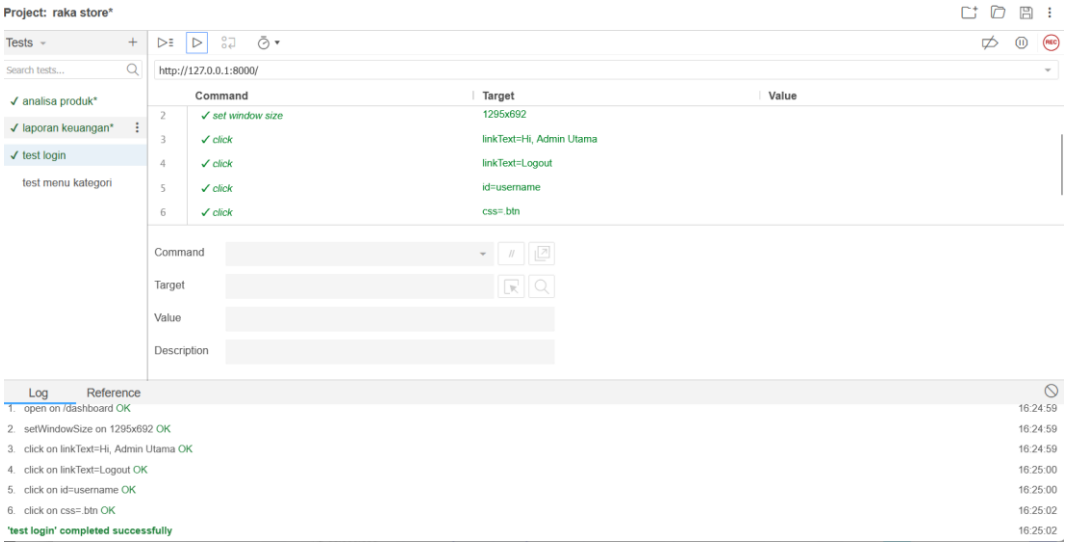
SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

Tegal, 15 Juli 2025
Dosen Pembimbing II

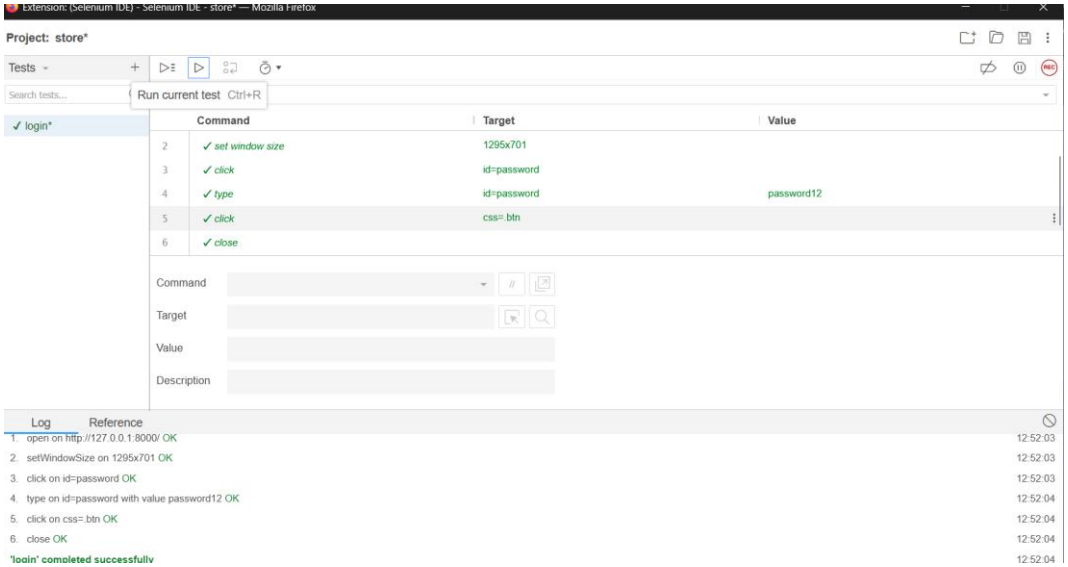
Sharfina Febbi Handayani, S.Kom., M.Kom.
NIPY. 08.020.451

Lampiran 7. *Selenium*

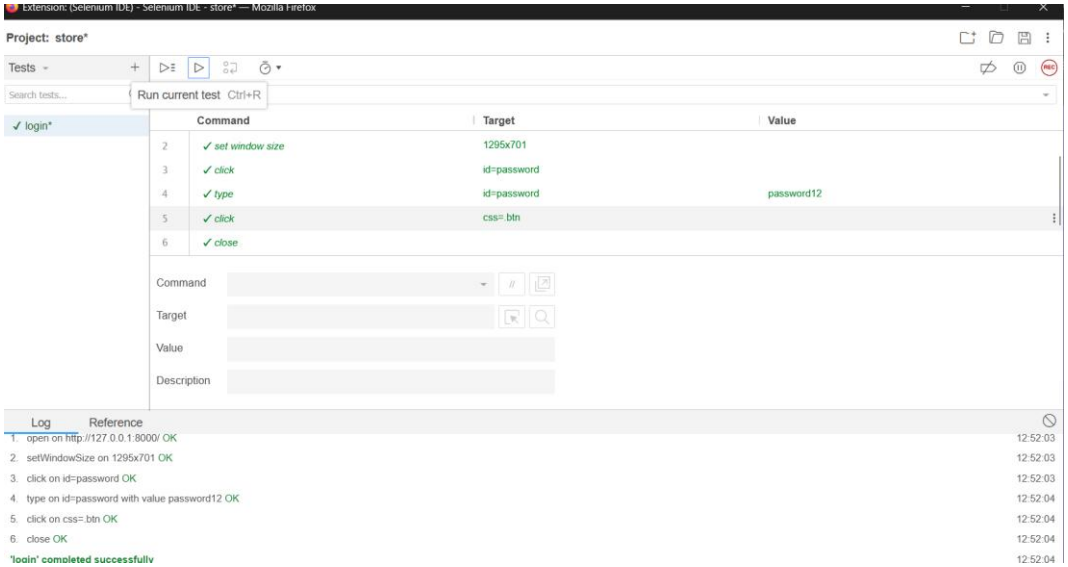
TC-001



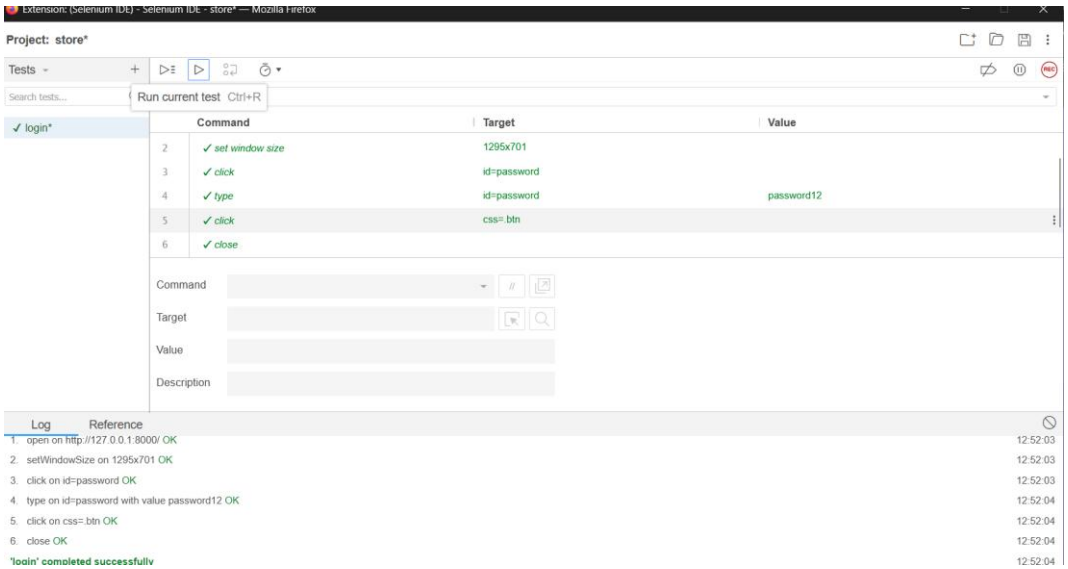
TC-002



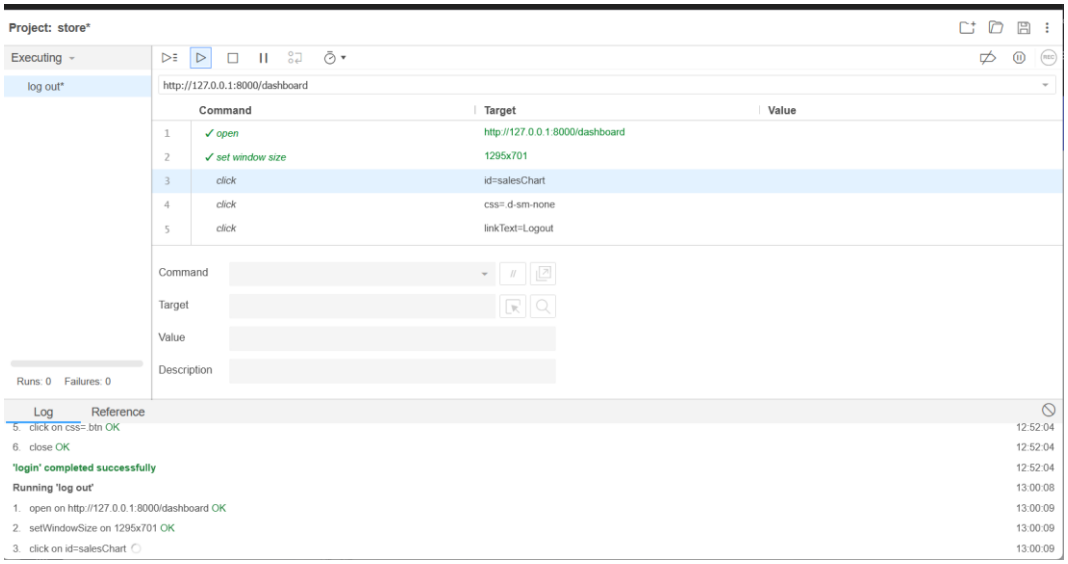
TC-003


Extension: (Selenium IDE) - Selenium IDE - store* — Mozilla Firefox
Project: store*
Tests
Run current test Ctrl+R
login*
Command
Target
Value
2 set window size 1295x701
3 click id=password
4 type id=password password12
5 click css=.btn
6 close
Command
Target
Value
Description
Log
Reference
1. open on http://127.0.0.1:8000/ OK 12:52:03
2. setWindowSize on 1295x701 OK 12:52:03
3. click on id=password OK 12:52:03
4. type on id=password with value password12 OK 12:52:04
5. click on css=.btn OK 12:52:04
6. close OK 12:52:04
'login' completed successfully 12:52:04

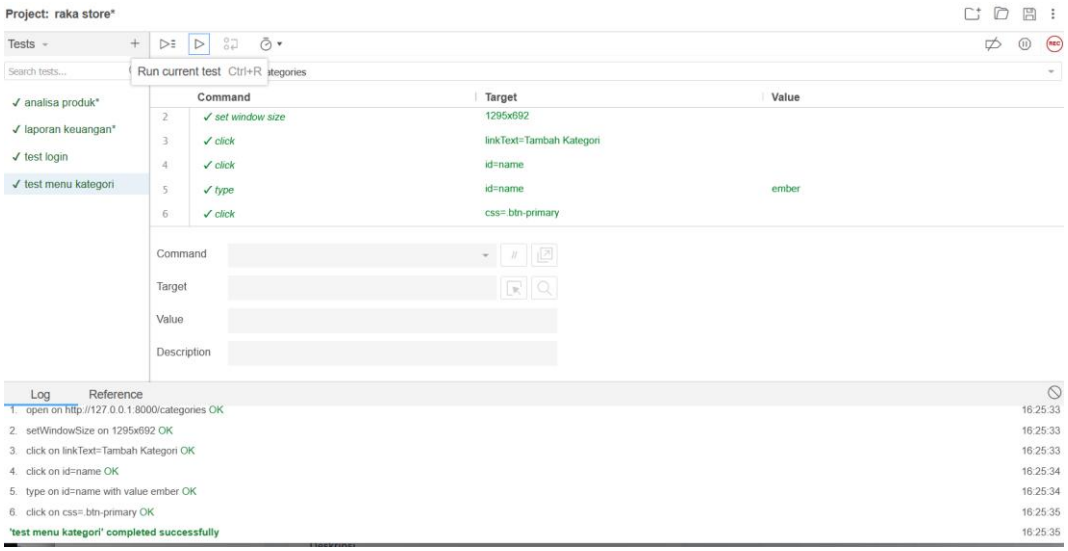
TC-004


Extension: (Selenium IDE) - Selenium IDE - store* — Mozilla Firefox
Project: store*
Tests
Run current test Ctrl+R
login*
Command
Target
Value
2 set window size 1295x701
3 click id=password
4 type id=password password12
5 click css=.btn
6 close
Command
Target
Value
Description
Log
Reference
1. open on http://127.0.0.1:8000/ OK 12:52:03
2. setWindowSize on 1295x701 OK 12:52:03
3. click on id=password OK 12:52:03
4. type on id=password with value password12 OK 12:52:04
5. click on css=.btn OK 12:52:04
6. close OK 12:52:04
'login' completed successfully 12:52:04

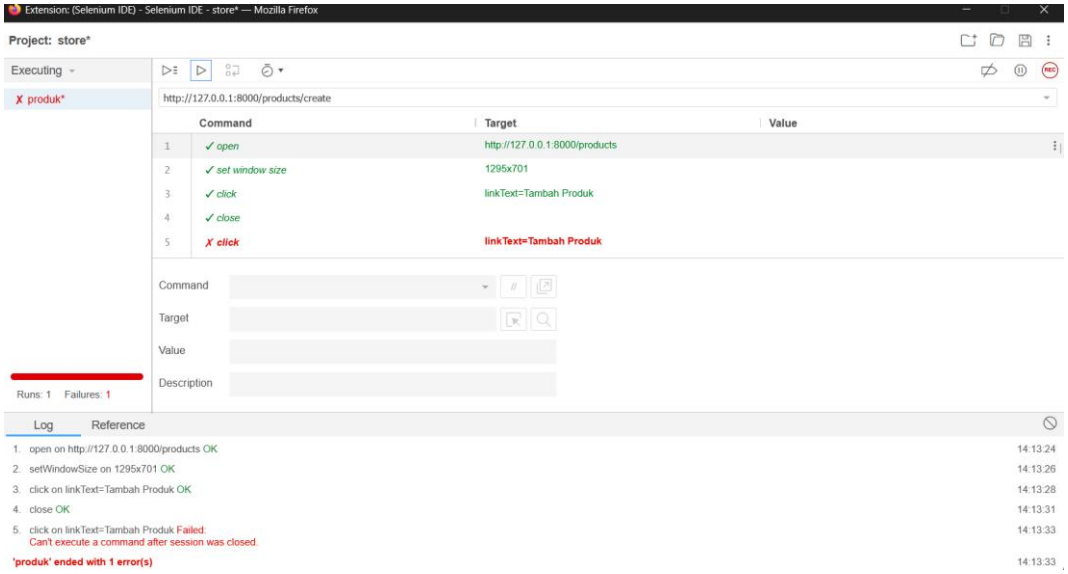
TC-005


Project: store*
Executing
log out*
http://127.0.0.1:8000/dashboard
Command
Target
Value
1 ✓ open http://127.0.0.1:8000/dashboard
2 ✓ set window size 1295x701
3 click id=salesChart
4 click css=.d-sm-none
5 click linkText=Logout
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
5. click on css=.btn OK 12:52:04
6. close OK 12:52:04
'login' completed successfully 12:52:04
Running 'log out' 13:00:08
1. open on http://127.0.0.1:8000/dashboard OK 13:00:09
2. setWindowSize on 1295x701 OK 13:00:09
3. click on id=salesChart 13:00:09

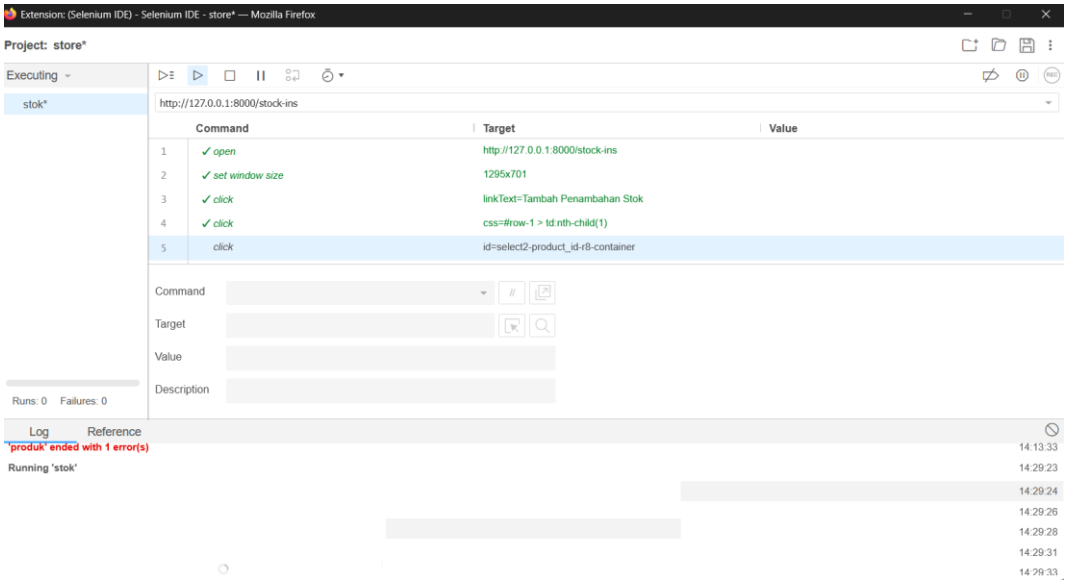
TC-006


Project: raka store*
Tests
Search tests...
Run current test Ctrl+R
✓ analisa produk*
✓ laporan keuangan*
✓ test login
✓ test menu kategori
Command
Target
Value
2 ✓ set window size 1295x692
3 ✓ click linkText=Tambah Kategori
4 ✓ click id=name
5 ✓ type id=name ember
6 ✓ click css=.btn-primary
Command
Target
Value
Description
Log
Reference
1. open on http://127.0.0.1:8000/categories OK 16:25:33
2. setWindowSize on 1295x692 OK 16:25:33
3. click on linkText=Tambah Kategori OK 16:25:33
4. click on id=name OK 16:25:34
5. type on id=name with value ember OK 16:25:34
6. click on css=.btn-primary OK 16:25:35
'test menu kategori' completed successfully 16:25:35

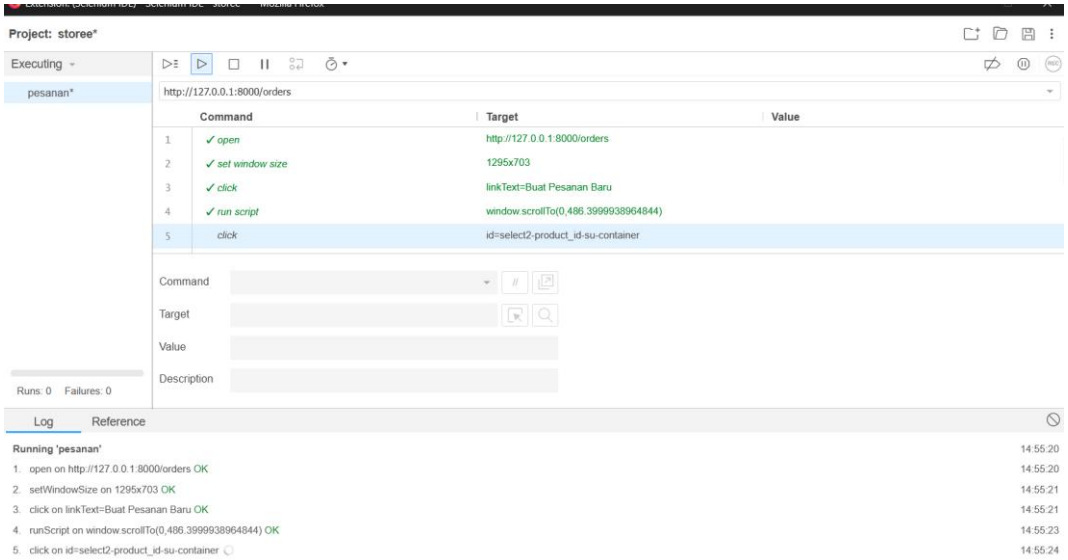
TC-007


Extension: (Selenium IDE) - Selenium IDE - store* — Mozilla Firefox
Project: store*
Executing
produk*
http://127.0.0.1:8000/products/create
Command
Target
Value
1 open http://127.0.0.1:8000/products
2 set window size 1295x701
3 click linkText=Tambah Produk
4 close
5 click linkText=Tambah Produk
Command
Target
Value
Description
Runs: 1 Failures: 1
Log
Reference
1. open on http://127.0.0.1:8000/products OK 14:13:24
2. setWindowSize on 1295x701 OK 14:13:26
3. click on linkText=Tambah Produk OK 14:13:28
4. close OK 14:13:31
5. click on linkText=Tambah Produk Failed: 14:13:33
Can't execute a command after session was closed.
'produk' ended with 1 error(s) 14:13:33

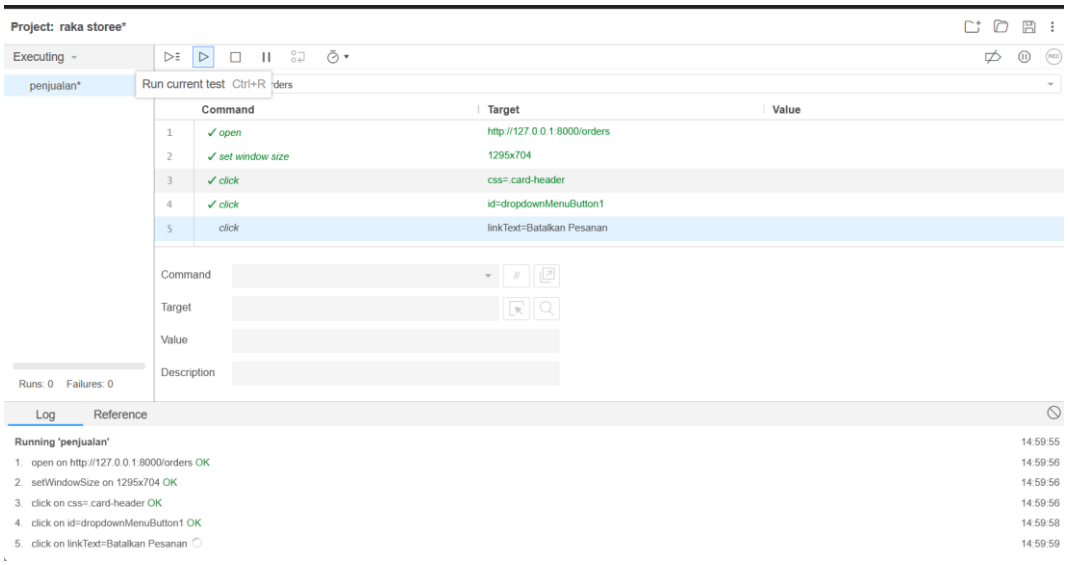
TC-008


Extension: (Selenium IDE) - Selenium IDE - store* — Mozilla Firefox
Project: store*
Executing
stok*
http://127.0.0.1:8000/stock-ins
Command
Target
Value
1 open http://127.0.0.1:8000/stock-ins
2 set window size 1295x701
3 click linkText=Tambah Penambahan Stok
4 click css=#row-1 > td:nth-child(1)
5 click id=select2-product_id-r8-container
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
'produk' ended with 1 error(s) 14:13:33
Running 'stok' 14:29:23
14:29:24
14:29:26
14:29:28
14:29:31
14:29:33

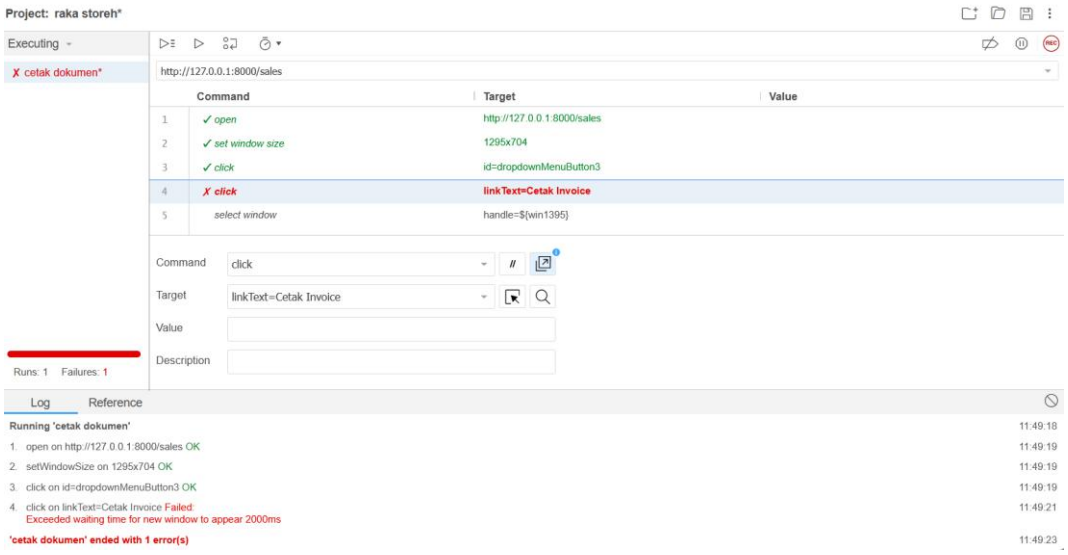
TC-009


Project: storee*
Executing
pesanan*
http://127.0.0.1:8000/orders
Command
Target
Value
1 ✓ open http://127.0.0.1:8000/orders
2 ✓ set window size 1295x703
3 ✓ click linkText=Buat Pesanan Baru
4 ✓ run script window.scrollTo(0,486.3999938964844)
5 click id=select2-product_id-su-container
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
Running 'pesanan'
1. open on http://127.0.0.1:8000/orders OK
2. setWindowSize on 1295x703 OK
3. click on linkText=Buat Pesanan Baru OK
4. runScript on window.scrollTo(0,486.3999938964844) OK
5. click on id=select2-product_id-su-container
14:55:20
14:55:20
14:55:21
14:55:21
14:55:23
14:55:24

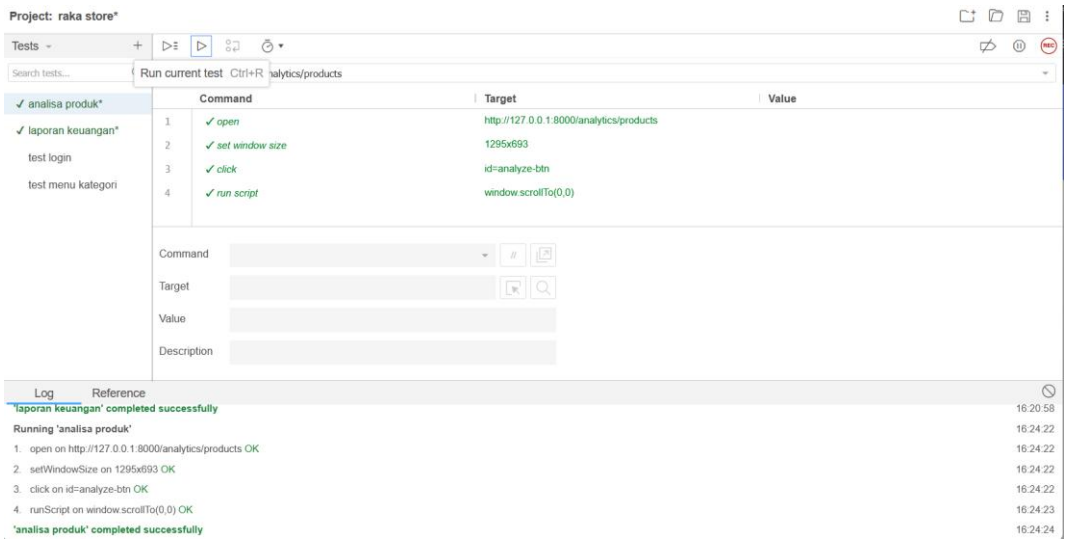
TC-010


Project: raka storee*
Executing
penjualan*
Run current test Ctrl+R
Command
Target
Value
1 ✓ open http://127.0.0.1:8000/orders
2 ✓ set window size 1295x704
3 ✓ click css=.card-header
4 ✓ click id=dropdownMenuButton1
5 click linkText=Batalkan Pesanan
Command
Target
Value
Description
Runs: 0 Failures: 0
Log
Reference
Running 'penjualan'
1. open on http://127.0.0.1:8000/orders OK
2. setWindowSize on 1295x704 OK
3. click on css=.card-header OK
4. click on id=dropdownMenuButton1 OK
5. click on linkText=Batalkan Pesanan
14:59:55
14:59:56
14:59:56
14:59:56
14:59:58
14:59:59

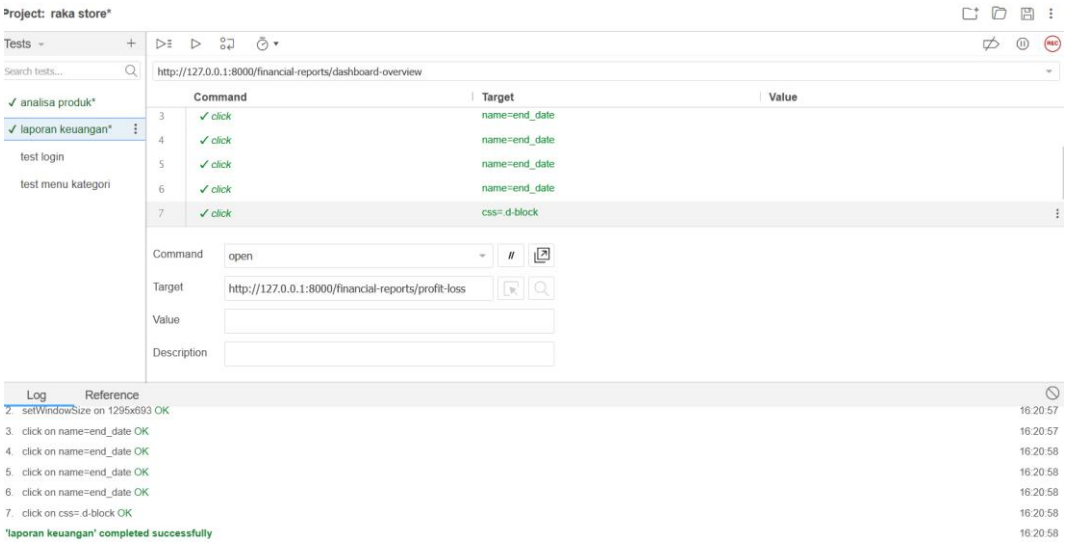
TC-011


Project: raka storeh*
Executing
✗ cetak dokumen*
http://127.0.0.1:8000/sales
Command
Target
Value
1 ✓ open http://127.0.0.1:8000/sales
2 ✓ set window size 1295x704
3 ✓ click id=dropdownMenuButton3
4 ✗ click linkText=Cetak Invoice
5 select window handle=${win1395}
Command click
Target linkText=Cetak Invoice
Value
Description
Runs: 1 Failures: 1
Log Reference
Running 'cetak dokumen' 11:49:18
1. open on http://127.0.0.1:8000/sales OK 11:49:19
2. setWindowSize on 1295x704 OK 11:49:19
3. click on id=dropdownMenuButton3 OK 11:49:19
4. click on linkText=Cetak Invoice Failed: 11:49:21
Exceeded waiting time for new window to appear 2000ms
'cetak dokumen' ended with 1 error(s) 11:49:23

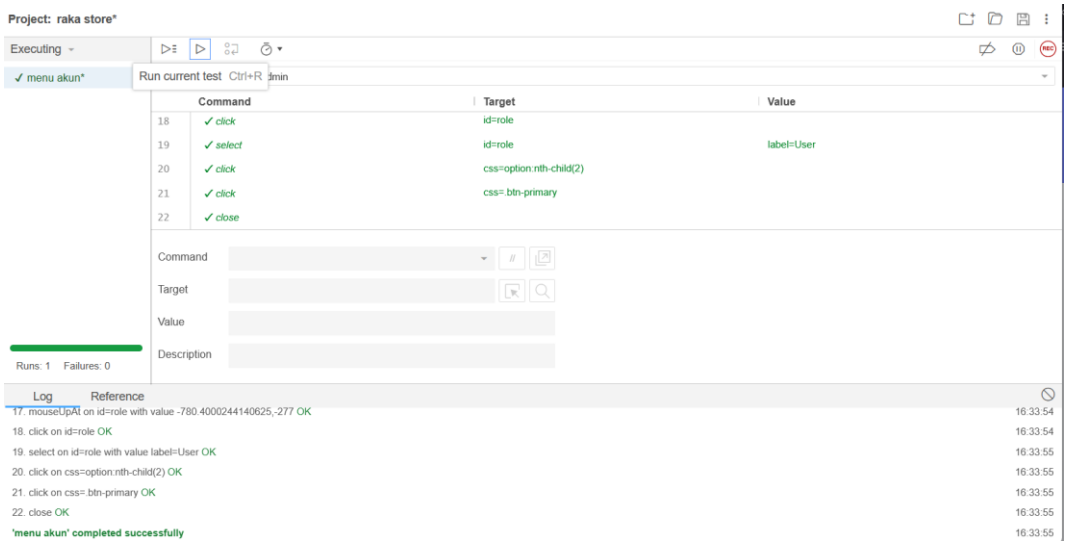
TC-012


Project: raka store*
Tests
Search tests...
Run current test Ctrl+R
✓ analisa produk*
✓ laporan keuangan*
test login
test menu kategori
Command
Target
Value
1 ✓ open http://127.0.0.1:8000/analytics/products
2 ✓ set window size 1295x693
3 ✓ click id=analyze-btn
4 ✓ run script window.scrollTo(0,0)
Command
Target
Value
Description
Log Reference
'laporan keuangan' completed successfully 16:20:58
Running 'analisa produk' 16:24:22
1. open on http://127.0.0.1:8000/analytics/products OK 16:24:22
2. setWindowSize on 1295x693 OK 16:24:22
3. click on id=analyze-btn OK 16:24:22
4. runScript on window.scrollTo(0,0) OK 16:24:23
'analisa produk' completed successfully 16:24:24

TC-013


Project: raka store*
Tests
Search tests...
✓ analisa produk*
✓ laporan keuangan*
test login
test menu kategori
http://127.0.0.1:8000/financial-reports/dashboard-overview
Command
Target
Value
3 ✓ click name=end_date
4 ✓ click name=end_date
5 ✓ click name=end_date
6 ✓ click name=end_date
7 ✓ click css=.d-block
Command open
Target http://127.0.0.1:8000/financial-reports/profit-loss
Value
Description
Log
Reference
2. setWindowSize on 1295x693 OK 16:20:57
3. click on name=end_date OK 16:20:57
4. click on name=end_date OK 16:20:58
5. click on name=end_date OK 16:20:58
6. click on name=end_date OK 16:20:58
7. click on css=.d-block OK 16:20:58
'laporan keuangan' completed successfully 16:20:58

TC-014


Project: raka store*
Executing
✓ menu akun*
Run current test Ctrl+R
Command
Target
Value
18 ✓ click id=role
19 ✓ select id=role label=User
20 ✓ click css=option:nth-child(2)
21 ✓ click css=.btn-primary
22 ✓ close
Command
Target
Value
Description
Runs: 1 Failures: 0
Log
Reference
17. mouseUpAt on id=role with value -780.4000244140625,-277 OK 16:33:54
18. click on id=role OK 16:33:54
19. select on id=role with value label=User OK 16:33:55
20. click on css=option:nth-child(2) OK 16:33:55
21. click on css=.btn-primary OK 16:33:55
22. close OK 16:33:55
'menu akun' completed successfully 16:33:55

Lampiran 8. *Tes case*

Project Name	Aplikasi Penjualan Berbasis *Web* Integrasi *Whatsapp Bot* Pada Raka *Store*								
Description	Testing Fungsi dari Aplikasi Penjualan Berbasis *Web* Integrasi *Whatsapp Bot* Pada Raka *Store*								
Modul Name	ID	Test Case	Pre-Contidion	Step	Test Data	Expected Result	Post Condisiton	Actual Result	Status
Login	TC-001	Memasukkan data yang benar	Harus sudah memiliki akun untuk login	1. Masukkan toko	Valid username	Akun berhasil login masuk ke halaman dasboard	Halaman dashboard terlihat	Menampilkan halaman dashboard	Pass
				2. Masukkan Password	Valid Password				
				3. Tekan Login button					
Login	TC-002	Memasukkan data password yang salah	Harus sudah memiliki akun untuk login	1. Masukkan toko	Valid username	Akun gagal masuk ke halaman dashboard	Halaman meminta memasukkan data yang benar	Menampilkan halaman login	Pass
				2. Masukkan Password12	Invalid Password				
				3. Tekan Login button					
Login	TC-003	Memasukkan data	Harus sudah memiliki	1. Masukkan toko	Valid username	Akun gagal masuk ke		Menampilkan halaman login	pass

		username yang salah	akun untuk login	2. Masukkan kalsibot	Invalid Password	halaman dashboard	Halaman meminta memasukkan data yang benar		
				3. Tekan Login button					
Login	TC-004	Memasukkan data yang salah	Harus sudah memiliki akun untuk login	1. Masukkan toko	Valid username	Akun gagal masuk ke halaman dashboard	Halaman meminta memasukkan data yang benar	Menampilkan halaman login	Pass
				2. Masukkan Password12	Invalid Password				
				3. Tekan Login button					
Logout	TC-005	Melakukan logout	Harus sudah login dalam dasboard	1. Klik Profil		Akun berhasil melakukan logout ke halaman login	Halaman menampilkan halaman login	Menampilkan halaman login	Pass
				2. Tekan Logout menu					
Kategori	TC-006	Melakukan penambahan data kategori	Harus sudah login dalam dasboard	1. Klik kategori		Akun berhasil membuka halaman tambah kategori dan mengisi data	Akun berhasil membuka halaman tambah kategori dan menampilkan form	Menampilkan halaman tambah kategori dan form kategori	Pass
				2. Tekan tambah kategori					
				Masukan cngkir					
Tambah produk	TC-007			1. Klik produk		Berhasil menampilkan		Menampilkan halaman tambah	vailed

		Melakukan penambahan produk	Harus sudah login dalam dasboard	2. Klik menu tambah produk		halaman tambah produk	Akun berhasil membuka halaman tambah produk dan menginputkan data	data produk dan berhasil disimpan	
				masukan cangkir plastic, min stok 1,harga beli Rp 12.000 ,harga jual Rp 15,000					
Tambah Data stok	TC-008	Melakukan tambah data stok	Harus sudah login dalam dasboard	1. Klik stok		Berhasil menampilkan halaman tambah data stok	Akun berhasil membuka halaman tambah data stok	Menampilkan halaman tambah data stok dan berhasil disimpan	skip
				2. Klik menu tambah data stok					
				Pilih cangkir plastik jumlah 11					
Tambah data pesanan	TC-009	Melakukan tambah data pesanan	Harus sudah login dalam dasboard	1.klik pesanan		Berhasil menampilkan halaman tambah data pesanan	Akun berhasil membuka halaman tambah pesanan dan menginputkan data	Menampilkan halaman tambah data pesanan dan berhasil disimpan	skip
				2. Klik menu tambah pesanan					

				Nama indra produk cangkir harga 15000					
Tambah data penjualan	TC-010	Melakukan tambah data penjualan	Harus sudah login dalam dasboard	1. Klik pesanan		Berhasil menampilkan halaman pensanan	Akun berhasil membuka halaman pesanan dan menginputkan data	menampilkan halaman pensanan	Pass
				2. Klik menu proses penjualan					
Cetak dokumen	TC-011	Melihat pesanan	Harus sudah login dalam dasboard	1. Klik aksi		Berhasil menampilkan halaman struk pesanan	Akun berhasil membuka halaman struk pesanan	Menampilkan halaman struk pesanan dan mencetak dokumen	vailed
				2. Klik menu cetak					
Analisa produk	TC-012	Melakukan Analisa produk	Harus sudah login dalam halaman dashboard	1. Input semua kolom parameter		Berhasil menampilkan halaman Analisa produk	Akun berhasil membuka halaman Analisa produk	Menampilkan halaman Analisa produk dan menampilkan hasil analisa	Pass
				2. Submit					
Melihat laporan keuangan	TC-013	Melihat laporan keuangban	Harus sudah login dalam	1. Klik laporan keuangan		Berhasil menampilkan dashboard	Akun berhasil membuka halaman dashboard laporan keuangan	Menampilkan halaman dashboard	Pass

			halaman dashboard	2. klik dashboard overview		laporan keuangan		laporan keuangan	
Tambah Akun	TC-014	Melakukan tambah data akun	Harus sudah login dalam halaman dashboard	1 Klik manajement akun		Berhasil menampilkan halaman tambah akun	Akun berhasil membuka halaman tambah akun dan menginputkan data	Menampilkan halaman manajemen akun dan berhasil disimpan	Pass
				2. Klik menu tamba akun					
				Nama indra role user Password password					